
Politecnico di Torino

**Politecnico di Torino**
Master Degree in Electronic Engineering

# Design Space Exploration of Integrated Circuit Floorplans through Area Minimization and ML-Guided Macro Placement

Candidate:
**Daniele Di Capua**

Supervisors:
**Prof. Guido Masera**
**Eng. Afonso Moreira**
**Eng. Cristiano Santos**

**Academic Year 2024-2025**

Politecnico di Torino
Dipartimento di Elettronica e Telecomunicazioni


Politecnico di Torino

Master Degree in Electronic Engineering

# Design Space Exploration of Integrated Circuit Floorplans through Area Minimization and ML-Guided Macro Placement

Candidate:
**Daniele Di Capua**

Supervisors:
**Prof. Guido Masera**
**Eng. Afonso Moreira**
**Eng. Cristiano Santos**

**Academic Year 2024-2025**

$A$

# Abstract

The increasing complexity of modern System-on-Chip (SoC) designs has made macro placement a critical yet largely manual task in the physical design flow. This thesis explores a Machine Learning (ML) based Electronic Design Automation (EDA) tool developed at Qualcomm® to automate and optimize macro placement during the floorplanning stage. The tool integrates a neural network engine to generate diverse macro placement alternatives, which are then evaluated using multi-objective metrics such as wirelength and congestion; a Graphical user interface (GUI) was developed to support interactive floorplan exploration, including area shrink optimization. The experimental campaign involved testing eight macro placements across four area configurations (0%, -1%, -2%, -3%) and analyzing their impact on Quality of Results (QoR) metrics such as utilization, timing, power, and design rule violations. Results show that moderate area reductions (up to -2%) can improve or preserve design quality compared to the default configuration, while aggressive compaction introduces significant risks. This work demonstrates the potential of ML-driven automation to enhance early-stage design exploration and support more informed architectural decisions in industrial Very Large-Scale Integration (VLSI) design flows.

# Contents

# List of Figures

# List of Tables

# Acronyms

**SoC** System-on-Chip

**EDA** Electronic Design Automation

**GUI** Graphical user interface

**QoR** Quality of Results

**VLSI** Very Large-Scale Integration

**AI** Artificial Intelligence

**ML** Machine Learning

**RL** Reinforcement Learning

**GNNs** Graph Neural Networks

**CNNs** Convolutional Neural Networks

**BO** Bayesian Optimization

**IC** Integrated circuit

**IP** Intellectual Property

**RTL** Register-Tranfer Level

**HDLs** Hardware Description Language

**ESD** Electrostatic Discharge

**DFM** Design For Manufacturability

**GANs** Generative Adversarial Networks

**QoR** Quality of Results

**DRC** Design Rule Checking

**LVS** Layout vs. Schematic

**ERC** Electrical Rule Checking

**KL** Kernighan-Lin

**FM** Fiducia-Mattheyses

**PLL** Phase locked loops

**DLL** Delay locked loops

**STA** Static timing analysis

**AAT** Actual arrival time

**RAT** Required arrival time

**TDP** Timing-driven placement

**WNS** Worst negative slack

**TNS** Total negative slack

**HPWL** Total half-perimeter wirelength

**WA** Weighted-Average

**PDE** Partial differential equation

**DCT** Discrete Cosine Transform

**DST** Discrete Sinusoidal Transform

**FFT** Fast Fourier Transform

**LEF** Library Exchange Format

**DEF** Design Exchange Format

**DC** direct-current

**HM** Hard Macro

**PnR** Place & Route

**ESD** Electrostatic Discharge

**PMUX** Power Mux

**PVT** Process-Voltage-Temperature

**TT** typical-typical

**SS** slow-slow

**FF** fast-fast

**PPA** Performance, Power and Area

**CCI** Company Confidential Information

**CTS** clock tree synthesis

# Introduction

The relentless growth in complexity of modern SoC designs, driven by Moore's Law and the increasing demand for performance, integration, and energy efficiency, has profoundly reshaped the landscape of semiconductor development. As transistor densities continue to rise and design sizes scale into the billions of components, the physical design phase of VLSI circuits has emerged as a critical bottleneck in the overall design flow. Among the various stages of the physical implementation process, macro placement plays a pivotal role in determining the quality and feasibility of the final layout; this task involves the positioning of large, pre-designed blocks such as memories, analog IPs, or custom logic within the chip floorplan, and has a direct impact on downstream stages including routing, timing closure, power distribution, and manufacturability. Despite its importance, macro placement remains one of the least automated and most intuition-driven stages of the physical design flow. Unlike standard cell placement, which is largely handled by mature commercial tools, macro placement is still predominantly performed manually by experienced physical design engineers; this process relies heavily on designer expertise, domain-specific heuristics, and iterative trial-and-error, making it time-consuming, error-prone, and difficult to scale. Moreover, the lack of automation in this phase significantly limits the ability to explore alternative floorplan configurations, especially in the early stages of design when architectural decisions are most impactful.

In recent years, the rapid advancement of Artificial Intelligence (AI) and ML has opened new avenues for innovation across the EDA landscape. These technologies have demonstrated remarkable potential in addressing long-standing challenges in digital design, offering data-driven alternatives to traditional heuristic-based methods. The motivation behind this thesis stems from the need to bridge the automation gap in macro placement by leveraging ML-based techniques to support and enhance early-stage floorplan exploration. By introducing intelligent, data-driven strategies into this critical phase, it becomes possible to generate diverse placement alternatives, evaluate them using multi-objective metrics, and guide designers toward more informed and efficient decisions. This approach not only improves productivity but also enables a more systematic exploration of the design space, ultimately contribut-

ing to higher-quality and more robust VLSI implementations.

Motivated by these considerations, this thesis investigates a novel approach to macro placement that integrates machine learning and interactive user control. At the core of this approach lies a proprietary macro placement tool developed at Qualcomm®, which serves as the foundation for the experimental work presented in this thesis. The tool leverages a neural network engine to generate diverse macro placement alternatives, exploring the design space efficiently; each candidate placement is evaluated using a set of multi-objective metrics, primarily total wirelength and routing congestion, which are critical indicators of downstream design quality.

To enhance usability and support early-stage architectural exploration, the tool has been extended with a GUI that enables designers to configure and visualize floorplan scenarios interactively. Through the GUI, users can define area shrink parameters, specifying the degree and direction of compaction (horizontal, vertical, bidirectional, or edge-specific), and set the number of placement candidates to be generated per configuration. This shrink-based exploration is particularly relevant in advanced technology nodes, where silicon area is at a premium and design robustness under tight constraints must be evaluated. Once the placements are produced, the GUI allows users to visually inspect the generated layouts and identify the most promising configurations, this enables a selective continuation of the design flow: high-quality placements can be promoted to subsequent implementation stages, while suboptimal ones can be discarded early, saving time and computational resources.

For each configuration, the tool generates a ranked set of candidate placements using a Pareto-based framework, balancing competing objectives without privileging a single metric. The top-ranked solutions are exported as `.tcl` scripts, ready to be sourced into commercial EDA tools, ensuring seamless integration with industrial design flows and enabling rapid prototyping and validation. By combining ML-driven generation, interactive configuration, visual inspection, and multi-objective evaluation, the proposed framework empowers designers to make more informed decisions early in the design flow.

This thesis is organized into three main chapters:

- Chapter 1 provides a comprehensive overview of the state of the art in physical design, with a particular focus on the placement phase and its challenges in the context of mixed-size designs. The chapter introduces the key steps of the VLSI design flow, discusses the limitations of current macro placement methodologies, and highlights the need for more scalable and automated solutions. Furthermore, the chapter includes an analysis of the state of the art in the application of machine learning within EDA tools, emphasizing how

ML techniques are increasingly being adopted to improve design efficiency, tackle complex optimization problems, and support the development of next-generation placement strategies.

- Chapter 2 presents the theoretical foundations of placement algorithms, including global placement, wirelength smoothing, density modeling, and nonlinear optimization. It also reviews existing analytical placers such as ePlace [1], ePlace-MS [3], and DREAMPlace [2], and introduces the proprietary macro placement tool developed at Qualcomm®. The chapter concludes with a detailed description of the tool's architecture and its integration into a floorplan exploration framework.

- Chapter 3 describes the experimental contribution developed during the internship; this includes: the extension of the macro placer with a GUI, the implementation of an area shrink optimization engine, and the execution of a test campaign involving eight macro placements across four area configurations. The results are analyzed using a wide range of physical and timing metrics, providing insights into the tool's effectiveness and limitations. Finally, the conclusions summarize the key findings of the thesis, reflect on the implications of the proposed approach, and outline potential directions for future work, including the integration of timing-driven placement and reinforcement learning techniques to further enhance automation and design quality.

Through this structure, the thesis aims to highlight how the integration of machine learning techniques and interactive tools contributes to automating macro placement, enhancing design space exploration, and supporting more informed architectural decisions in the early stages of the physical design flow.

# Chapter 1

# State of Art

As Integrated circuit (IC) become increasingly complex and performance requirements more stringent, the physical implementation of digital systems presents a growing set of technical and methodological difficulties. The purpose of this chapter is to provide a comprehensive overview of the current challenges in digital circuit design, with a particular emphasis on the physical design phase.

The chapter begins, in section 1.1, by revisiting Moore's Law, a long-standing prediction that has guided the semiconductor industry for decades. This section explores the historical relevance of Moore's Law, its impact on technological progress, and the growing concerns about its continued applicability in the modern era.

Section 1.2 then analyzes the state of the art in the application of ML within EDA tools, highlighting recent innovations and research trends that are shaping the future of VLSI design.

Following this, in the following paragraph 1.3, the chapter outlines the key steps involved in the design of a VLSI circuit, offering a high-level view of the design flow from specification to implementation. This provides the necessary context for understanding the intricacies of the design process.

The final section 1.4 delves deeper into the physical design phase, examining each step in detail. This includes partitioning, chip planning, placement, clock tree synthesis, routing and timing closure, highlighting the technical challenges and trade-offs encountered during this critical stage of the design cycle.

Through this structure, the chapter aims to set the foundation for the subsequent discussions in the thesis by framing the technological landscape and design methodologies that shape modern digital circuit development. In particular, it lays the groundwork for the analysis of a specific limitation within the current physical design flow: the difficulty of achieving innovative and efficient macro placement during the placement phase. Despite the increasing complexity of modern SoCs, macro

placement remains largely a manual and heuristic-driven process, heavily reliant on designer experience and intuition. This lack of automation not only limits design space exploration but also poses significant challenges in terms of scalability and optimization. By highlighting this issue early on, the chapter prepares the reader for a deeper investigation into potential solutions and methodologies aimed at improving macro placement strategies in the later sections of the thesis.

## 1.1 Moore's Law

In his seminal 1965 article [4], Gordon E. Moore outlined a visionary perspective on the future of integrated electronics, proposing that the number of components on a silicon chip would continue to increase at a steady, exponential rate. Rather than focusing on a specific numerical target, Moore emphasized the trend of rapid growth in integration density, driven by advancements in photolithography, materials, and manufacturing techniques. He argued that this trend would lead to significant improvements in cost-efficiency, reliability, and performance, enabling a wide range of applications, from home computing to automotive automation and portable communication devices.

Moore also identified several technical challenges that could hinder this progress, such as heat dissipation, interconnect delays, and manufacturing yields. However, he maintained that these obstacles could be overcome primarily through engineering innovation rather than fundamental scientific breakthroughs. His insights laid the foundation for what would later become known as **Moore's Law**: the empirical observation that the number of transistors on an IC doubles approximately every two years, as shown in Fig. 1.1. This principle has since become a cornerstone of the semiconductor industry, guiding long-term planning and setting ambitious targets for research and development.

Over the decades, Moore's Law has driven exponential growth in computing power, enabling the miniaturization and performance enhancements that define modern electronics. Despite growing concerns about physical and economic limitations, such as quantum effects, power density, and the rising cost of advanced fabrication nodes, ongoing research continues to push the boundaries of what is possible; innovations in 3D integration, new materials, and design automation aim to sustain the spirit of Moore's prediction, even as traditional scaling slows.

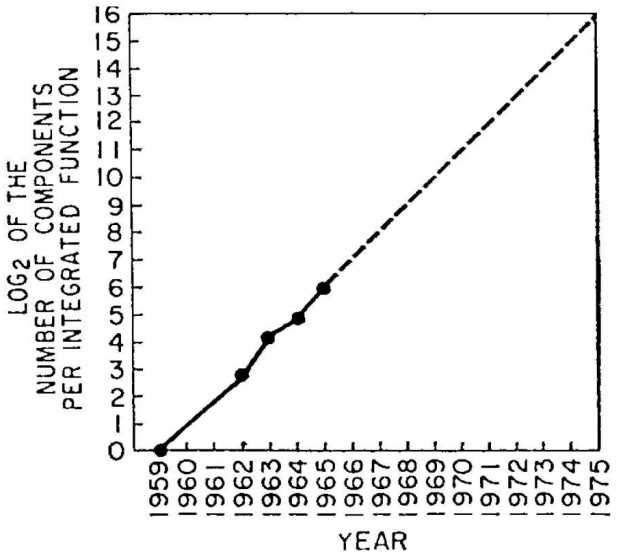



Figure 1.1: Moore's Law.

Given the increasing complexity and scale of modern SoCs, the physical design phase has become a critical bottleneck, particularly in tasks such as macro placement, which remain largely heuristic-driven and dependent on designer expertise. In response to these challenges, the EDA community has increasingly turned to data-driven and machine learning approaches to enhance automation, efficiency, and solution quality throughout the design flow.

## 1.2 Machine Learning in EDA Tools

In recent years, the integration of ML techniques into EDA tools has emerged as a transformative trend, aiming to address the escalating complexity, design costs, and time-to-market pressures in IC development. ML methods are now being explored and deployed across nearly all stages of the EDA flow, from high-level synthesis and logic optimization to physical design, verification, and manufacturing. The motivations for this shift are multiple: ML models can learn from vast historical design data, capture intricate dependencies that elude traditional heuristics, and provide rapid predictions or optimizations that would otherwise require time-consuming simulations or manual intervention [5] [6].

Within the physical design domain, ML applications are particularly prominent in tasks such as placement, routing, congestion prediction, power estimation, and design space exploration. ML models are typically used in four main roles [5]:

1. **decision making in traditional methods**, where ML replaces brute-force or empirical parameter tuning (e.g., selecting tool configurations or optimization strategies);

2. **performance prediction**, where supervised models (e.g., Regression, Random Forests, Convolutional Neural Networks (CNNs), Graph Neural Networks (GNNs)) are trained to estimate key metrics such as wirelength, congestion, timing, or power based on features extracted from netlists or layouts;

3. **black-box optimization**, where ML-guided surrogate models (e.g., Gaussian Processes, Random Forests) accelerate the search for optimal design points in large, expensive-to-evaluate spaces;

4. **automated design**, where advanced techniques like deep Reinforcement Learning (RL) and GNNs are used to directly generate or refine design solutions, such as macro and standard cell placements.

A landmark example of ML-driven automation is Google's RL-based macro placement framework, which models chip floorplanning as a sequential decision-making problem. Using a graph convolutional neural network to encode the netlist and a policy network trained via RL, their system can generate manufacturable floorplans for large chips in under six hours, orders of magnitude faster than traditional manual approaches, while achieving Performance, Power and Area (PPA) metrics comparable or superior to human experts [7]. The RL agent benefits from transfer learning: pre-training on a diverse set of chip blocks enables rapid adaptation and high-quality results on new, unseen designs, even in "zero-shot" mode [7]. This approach has been successfully deployed in production and has inspired a wave of research into RL and GNN-based placement and optimization methods [5] [7].

Beyond RL, other ML paradigms are also gaining traction. For example, Bayesian Optimization (BO) has been proposed as a sample-efficient alternative to RL and simulated annealing for macro placement. BO uses a probabilistic surrogate model to guide the search over combinatorial spaces (e.g., sequence pairs representing macro orderings), balancing exploration and exploitation to minimize objectives such as Total half-perimeter wirelength (HPWL) or congestion. BO has demonstrated competitive or superior performance to simulated annealing on standard benchmarks, with far fewer expensive evaluations, and is particularly attractive when the true objective is costly to compute [8].

Despite these advances, several challenges remain before ML-based macro placement and physical design tools can achieve widespread industrial adoption. First, **generalization** is a key concern: ML models often struggle to transfer knowledge across different technology nodes, design styles, or constraint sets, necessitating large, high-quality datasets and careful feature engineering [5] [9]. Second, **integration** with existing EDA flows is non-trivial, as ML-generated placements must be compatible with downstream tools and constraints, and must produce repeatable, explainable results[9]. Third, **reproducibility** and **stability** are essential for

industrial deployment: ML methods must deliver consistent outcomes across runs and support incremental design changes without requiring full retraining or manual intervention [9]. Finally, there is a growing emphasis on **trusted** ML, ensuring that models are reliable, fair, and interpretable for designers [5].

In summary, ML is rapidly reshaping the landscape of EDA, offering new paradigms for automation, prediction, and optimization in physical design. While significant progress has been made, continued research is needed to address issues of generalization, integration, and trust, and to fully realize the promise of ML-powered EDA tools in industrial practice.

To understand where these innovations are most impactful, and how they interact with established engineering practices, it is crucial to examine the structure of the VLSI design flow itself. The following section provides a comprehensive overview of this flow, outlining its key stages and interdependencies, and establishing the foundation for a more detailed discussion of the physical design phase, where both conventional and machine learning-driven methods converge to address some of the most significant bottlenecks in contemporary IC development.

## 1.3 VLSI Design Flow

Building upon the historical context and the challenges outlined earlier, it becomes clear that a structured and methodical approach is essential to manage the complexity of modern IC design. The VLSI design flow provides such a framework, guiding the transformation of a high-level functional specification into a fully verified and manufacturable chip. This flow is composed of several interdependent stages, each addressing specific aspects of the design process: from initial behavioral modeling to physical implementation and final verification. Understanding these steps is crucial not only for appreciating the overall design methodology but also for identifying where critical challenges, such as those in physical design, tend to emerge.

The design of a VLSI circuit is a highly intricate process that can be broken down into several distinct steps. These steps range from high-level system specifications to detailed physical design and verification before fabrication [10]. The major steps in the VLSI design flow are illustrated in Fig. 1.2 and discussed in detail below.

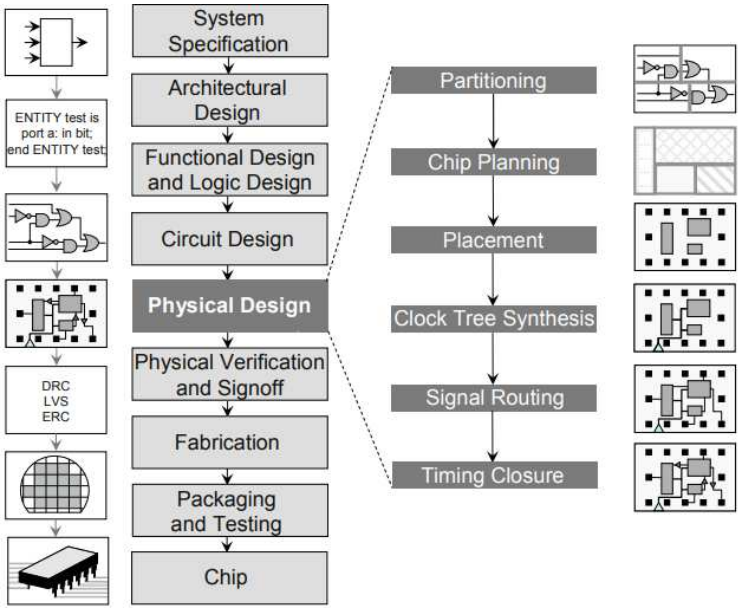



Figure 1.2: Main steps of the VLSI desgin flow.

- **System Specification**
  The initial phase involves defining the overall goals and high-level requirements of the system. This is a collaborative effort among chip architects, circuit designers, product marketers, operations managers, and layout and library designers. The goals and requirements encompass functionality, performance, physical dimensions, and production technology.

- **Architectural Design**
  In this phase, a basic architecture is determined to meet the system specifications. Key decisions include the integration of analog and mixed-signal blocks, memory management, the number and types of computational cores, internal and external communication protocols, and the usage of Intellectual Property (IP) blocks.

- **Functional and Logic Design**
  Once the architecture is set, the functionality and connectivity of each module must be defined. Functional design focuses on the high-level behavior of each module, while logic design is performed at the Register-Tranfer Level (RTL) using Hardware Description Language (HDLs) like Verilog and VHDL. Logic synthesis tools then convert HDL descriptions into low-level circuit elements.

- **Circuit Design**
  For the bulk of digital logic, logic synthesis tools automatically convert Boolean expressions into a gate-level netlist. However, critical low-level elements such as static RAM blocks, I/O, analog circuits, high-speed functions, and Electrostatic

Discharge (ESD) protection circuits are designed at the transistor level. Circuit simulation tools like SPICE verify the correctness of these designs.

- **Physical Design**
  Physical design involves instantiating all design components with their geometric representations. This includes assigning spatial locations (placement) and completing routing connections in metal layers. The result is a set of manufacturing specifications that must be verified. Physical design is performed with respect to design rules that represent the physical limitations of the fabrication medium.

- **Physical Verification and Signoff**
  After physical design, the layout must be fully verified to ensure correct electrical and logical functionality. This includes:

  1. ***Design Rule Checking (DRC):*** Verifies that the layout meets all technology-imposed constraints.
  2. ***Layout vs. Schematic (LVS) Checking:*** Ensures the functionality of the design by comparing the layout-derived netlist with the original netlist.
  3. ***Parasitic Extraction:*** Derives electrical parameters from the layout elements to verify the circuit's electrical characteristics.
  4. ***Antenna Rule Checking:*** Prevents antenna effects that may damage transistor gates during manufacturing.
  5. ***Electrical Rule Checking (ERC):*** Verifies the correctness of power and ground connections and ensures signal transition times, capacitive loads, and fanouts are appropriately bounded.

Among the various stages that compose the VLSI design flow just been discussed, the physical design phase plays a pivotal role in translating the logical representation of a circuit into a geometrical layout that can be fabricated on silicon. While earlier steps focus on functional correctness and architectural decisions, physical design is where spatial, timing, and manufacturing constraints converge, making it one of the most complex and constraint-driven parts of the flow. Given its critical importance and the unique challenges it presents, especially in the context of modern, large-scale systems, this phase warrants a more detailed examination. The following section delves into each step of the physical design process, highlighting the methodologies, tools, and design considerations involved.

## 1.4 Physical Design

Physical design of integrated circuits is a crucial and complex aspect of EDA. It involves the placement and routing of, nowadays, billions of transistors on a silicon chip, ensuring that they are interconnected efficiently and meet performance requirements. As technology advances, the number of transistors on a chip continues to grow, necessitating sophisticated algorithms to manage this complexity. One of the primary challenges in physical design is managing the delays caused by the interconnecting wires. In the past, achieving timing goals was largely dependent on the optimal placement of devices while with modern designs, timing constraints can only be verified after the final routing is completed, making the process more intricate. The continuous increase in transistor count and the growing interdependence between physical, timing, and logic domains require a fresh approach to the fundamental algorithms of chip implementation. Modern physical design flows must address multi-objective optimization, integrating various stages from design partitioning and floorplanning to electrical rule checking. This evolving landscape demands that experts in specific areas, such as routing or Design For Manufacturability (DFM), understand the broader implications of their work on the entire design flow. Physical design remains a dynamic field, continually pushing the boundaries of what is possible in semiconductor technology, and it plays a pivotal role in the advancement of chip design tools and methodologies [10].

In the remainder of this chapter, the individual steps that constitute the physical design process, shown in Fig. 1.2, will be examined in detail. Each phase will be discussed with particular attention to its functional objectives, the constraints it must satisfy, and the methodologies and tools typically employed in its execution. This in-depth analysis aims to provide a clearer understanding of how logical circuit descriptions are systematically transformed into manufacturable layouts, and to highlight the critical challenges that arise throughout this phase of the VLSI design flow.

### 1.4.1 System Partitioning

The complexity of modern IC designs has escalated to an unprecedented level, rendering tasks such as full-chip layout increasingly challenging. A prevalent approach involves partitioning the design into smaller segments, allowing for independent processing and parallel execution. This divide-and-conquer methodology can be applied by individually laying out each block and subsequently reassembling the outcomes as geometric partitions. While this strategy was historically utilized for manual partitioning, it has become impractical for extensive netlists. However, manual partitioning can still be executed within the framework of system-level modules, treating

them as singular entities when hierarchical information is accessible. Conversely, automated netlist partitioning can effectively manage large netlists and redefine the physical hierarchy of an electronic system, encompassing everything from boards to chips and from chips to blocks. Furthermore, traditional netlist partitioning can be advanced to multilevel partitioning, which is applicable for managing large-scale circuits.

A widely accepted method for reducing the design complexity of contemporary integrated circuits involves dividing them into smaller modules. The partitioning process separates the circuit into multiple subcircuits (partitions or blocks) while aiming to minimize the interconnections between these partitions, adhering to design constraints such as maximum partition sizes and allowable path delays. If each block is developed independently, without regard for other partitions, the connections between these blocks may adversely impact the overall design performance, leading to increased circuit delays or diminished reliability. Additionally, a high number of interconnections between partitions can create inter-block dependencies that hinder design efficiency. Consequently, the main objective of partitioning is to segment the circuit in a manner that reduces the number of connections between subcircuits, as shown in Fig. 1.3 Each partition must also comply with all design specifications.

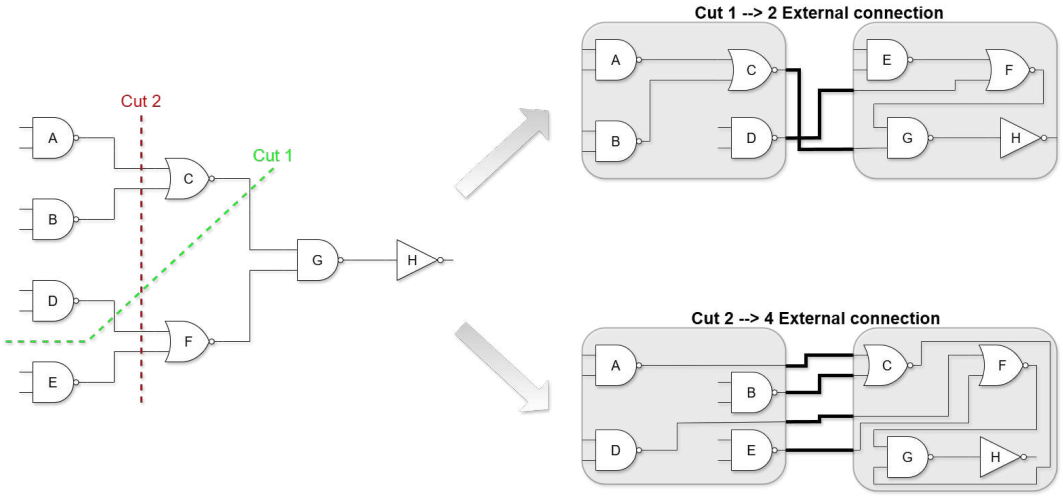



Figure 1.3: Example of partitioning.

Circuit partitioning is classified as NP-hard, this means that as the size of the problem increases linearly, the resources required to identify an optimal solution escalate at a rate surpassing any polynomial function. Currently, there is no established polynomial-time algorithm that guarantees a globally optimal solution for balance-constrained partitioning. Nevertheless, several effective heuristics were introduced during the 1970s and 1980s. These algorithms yield high-quality circuit partitioning results and are typically executed in low-order polynomial time, including the

Kernighan-Lin (KL) algorithm [11] and the Fiducia-Mattheyses (FM) algorithm [12]. Furthermore, optimization through simulated annealing can be employed to tackle particularly challenging partitioning problems. Generally, stochastic hill-climbing algorithms necessitate more than polynomial time to generate high-quality solutions, although they can be expedited at the cost of solution quality. In practice, simulated annealing seldom proves to be competitive.

### 1.4.2 Chip Planning

Chip planning involves the organization of substantial components such as caches, embedded memories, and IP cores, which possess defined areas, either fixed or variable shapes, and potentially designated locations. In instances where modules are not explicitly defined, chip planning utilizes netlist partitioning (Sec. 1.4.1) to discern these modules within extensive designs. The process of assigning shapes and locations to circuit modules during chip planning results in the formation of blocks, facilitating preliminary assessments of interconnect length, circuit delay, and overall chip performance (Fig. 1.4).

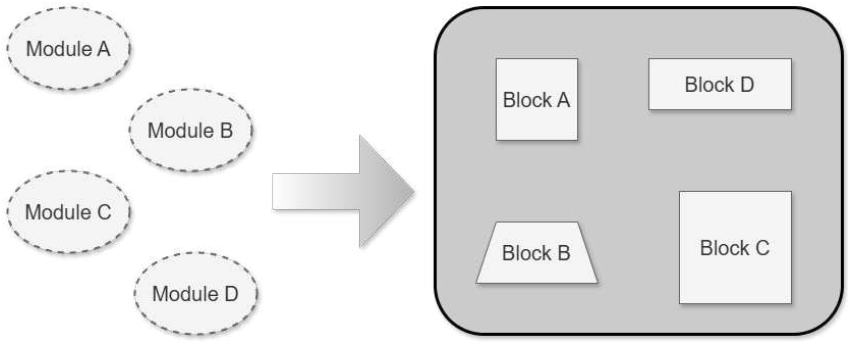



Figure 1.4: A module represents a collection of logic within a defined area. Once it is given a specific shape or dimensions, it is referred to as a block.

This initial analysis can highlight modules that require enhancements. The chip planning process is divided into the following phases:

1. *Floorplanning*
   Prior to the floorplanning phase, the design is divided into separate circuit modules. Each module is transformed into a rectangular block once it receives specific dimensions or a defined shape. These blocks can be categorized as either hard or soft. Hard blocks have fixed dimensions and areas, whereas soft blocks maintain a constant area but allow for variations in aspect ratio, which can be adjusted continuously or in discrete increments. The complete arrangement of these blocks, along with their respective positions, is referred to as a floorplan.

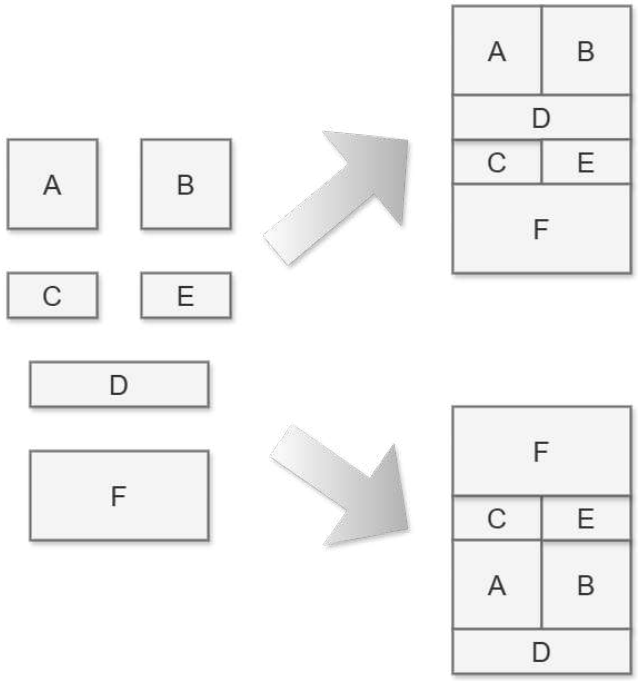



Figure 1.5: Example of two possible floorplans for the same set of blocks.

In extensive designs, individual modules may also undergo floorplanning in a recursive top-down manner; however, it is typical to concentrate on one hierarchical level of floorplanning at a time. In this context, the floorplan at the highest level is designated as the top-level floorplan. The floorplanning phase is crucial as it ensures that each chip module is allocated a shape and a location to facilitate gate placement, and every pin with an external connection is assigned a location to enable the routing of internal and external nets. This stage establishes the external attributes, fixed dimensions and external pin locations, for each module, which are essential for the subsequent placement and routing processes that define the internal characteristics of the blocks. Floorplan optimization encompasses various degrees of freedom; while it incorporates certain elements of placement and connection routing, the optimization of module shapes is distinct to floorplanning. The use of hard blocks in floorplanning is particularly significant when reusing existing blocks, including IP. From a mathematical perspective, this issue can be interpreted as a constrained scenario of floorplanning with soft parameters, although in practice, it may necessitate specialized computational methods.

Despite the critical role that floorplanning plays in shaping the overall quality and feasibility of a VLSI design, this phase remains, to a large extent, a manual and experience-driven process. As a result, it is still challenging to conduct extensive and systematic studies aimed at exploring alternative floorplan

configurations; particularly those that might reduce chip area or improve performance by adopting non-conventional macro placements within a block. The lack of automation and flexibility in current methodologies limits the ability to assess whether better results could be achieved by deviating from standard practices. To address this gap, Chapter 3 of this thesis focuses on the development of a tool designed to facilitate floorplan exploration. This tool enables designers to efficiently experiment with multiple area configurations and macro placement strategies within a given block. By integrating an internal machine learning engine, the tool can automatically generate a wide range of plausible macro placements for each area, significantly expanding the design space that a single engineer can explore. This approach aims to enhance productivity, support data-driven decision-making, and ultimately contribute to more optimized and innovative floorplan solutions.

2. *Pin Assignement*
   Due to the substantial geometric dimensions of blocks in floorplanning, the positioning of terminal locations for nets that interconnect these blocks is crucial. Typically, I/O pins are situated at the edges of a block to minimize interconnect length. Nevertheless, the optimal locations are contingent upon the relative arrangement of the blocks.In the process of pin assignment, each net is allocated to distinct pin locations to enhance the overall performance of the design. Key optimization objectives often include maximizing routability and minimizing electrical parasitics both within and beyond the block.

3. *Power Planning*
   The scaling of on-chip supply voltages occurs at a slower rate compared to chip frequencies and transistor counts. Consequently, the currents delivered to the chip progressively rise with each technological advancement. Enhanced packaging and cooling solutions, along with market demands for increased functionality, result in larger power budgets and more compact power grids. Currently, approximately 20-40% of all metal resources on the chip are allocated for power (VDD) and ground (GND) networks. Given that floorplanning is a precursor to place-and-route, power-ground planning has emerged as a critical component of contemporary chip design. Chip planning not only dictates the configuration of the power-ground distribution network but also influences the positioning of supply I/O pads (in wire-bond packaging) or bumps (in flip-chip packaging) (Fig. 1.6). These pads or bumps are strategically placed in or near regions of high activity on the chip to reduce the IR voltage drop.

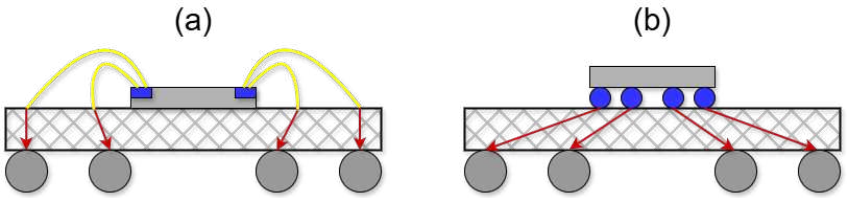



Figure 1.6: (a) wire bonding and (b) flip-chip packaging.

The power planning process is generally iterative, involving:

(a) initial simulations of significant power dissipation elements,

(b) preliminary assessments of chip power,

(c) evaluations of total chip power and peak power density,

(d) examinations of total chip power variations,

(e) investigations of inherent and additional fluctuations caused by clock gating,

(f) early analyses of power distribution, including average, maximum, and multi-cycle fluctuations.

To develop an effective supply network, various design and process technology factors must be taken into account. For instance, to accurately estimate chip power, the designer should consider:

(a) the implementation of low-Vth devices and dynamic circuits that have higher power consumption,

(b) the application of clock gating to reduce power usage,

(c) the quantity and strategic placement of additional decoupling capacitors to alleviate switching noise.

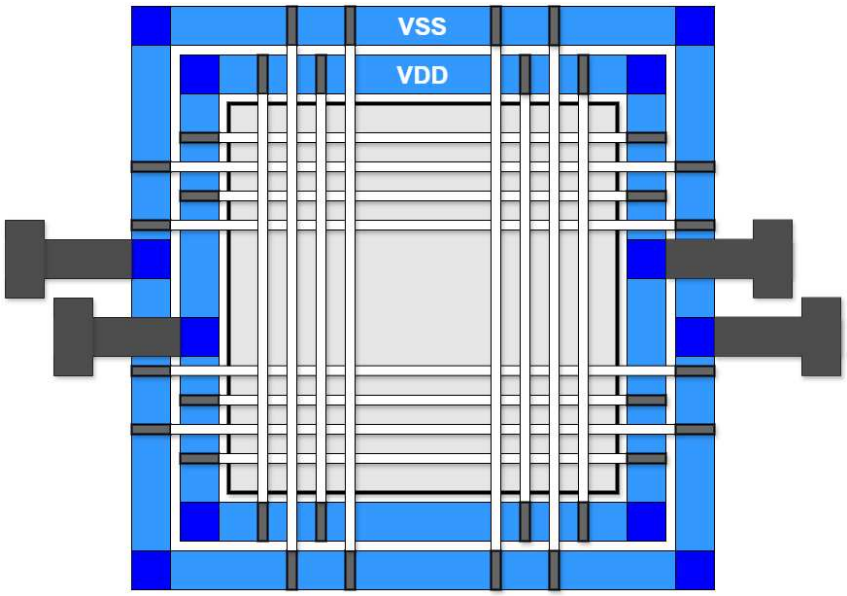



Figure 1.7: Power-ground routing in modern digital ICs typically has a mesh topology composed of: rings, I/O pads, stripes.

### 1.4.3 Placement

Placement aims to identify the positions of standard cells or logic components within each block, while also focusing on optimization goals such as reducing the overall length of connections between these elements. In particular, *global placement* (Fig. 1.9) designates general positions for movable objects, whereas *detailed placement* fine-tunes these positions to conform to legal cell sites and ensures that there are no overlaps. The precise locations obtained through detailed placement facilitate more accurate assessments of circuit delay, which is essential for timing optimization. Global placement frequently overlooks the specific shapes and dimensions of placeable objects, failing to align their positions with appropriate grid rows and columns. Some degree of overlap between placed objects is permitted, as the focus is on effective global positioning and the overall distribution of density. Legalization occurs either prior to or during the detailed placement phase, aiming to align placeable objects with the grid while eliminating overlaps, all the while striving to minimize displacements from their global placement positions and reducing the effects on interconnect length and circuit delay.

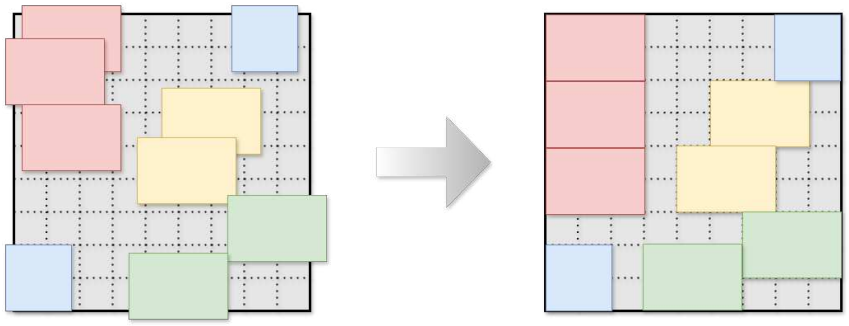


Figure 1.8: Example of legalization process: overlaps are removedand objects are aligned with the grid.

Detailed placement enhances the positioning of each standard cell through local adjustments, such as swapping two objects or shifting multiple objects in a row to accommodate another object. While global and detailed placement generally exhibit similar runtimes, global placement often demands significantly more memory and presents greater challenges for parallelization.

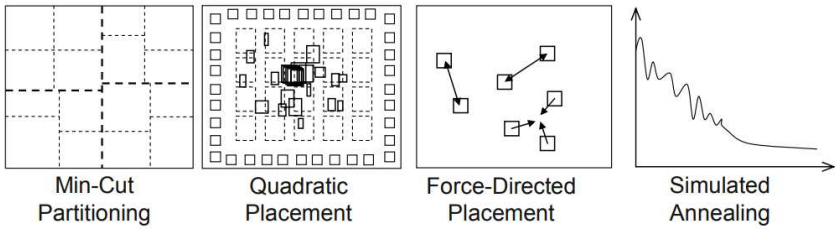



Figure 1.9: Techniques for global placement.

The placement process must create a layout that allows for the simultaneous routing of all design nets, ensuring that the placement is routable. Furthermore, it is essential to consider electrical factors such as signal delay and crosstalk. Since detailed routing information is not accessible during the placement phase, the placer focuses on optimizing estimates of routing quality metrics, including total weighted wirelength, cut size, wire congestion (density), and maximum signal delay. Given that the delay of a net is directly related to its length, placers frequently aim to reduce the *overall wirelength*.

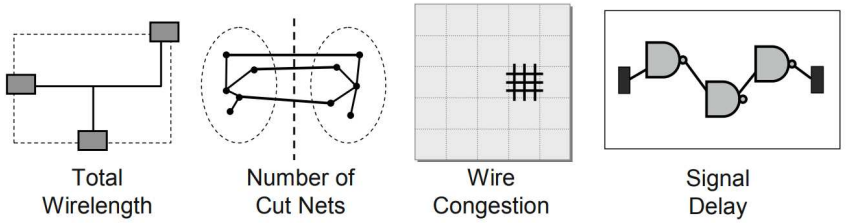



Figure 1.10: Example of routing metrics.

The automation of standard cell placement is essential for effectively handling the scale and complexity of contemporary VLSI designs, which encompass billions of components. Consequently, placement tools such as Cadence Innovus and Synopsys IC Compiler are extensively utilized within the VLSI industry. In contrast, macro placement is predominantly performed manually by physical design engineers, who adhere to heuristic guidelines derived from their experience. This manual process is labor-intensive and monotonous, limiting the potential for comprehensive floorplan exploration to identify superior solutions. Despite its significance as the foundation for subsequent physical design processes, macro placement greatly impacts the quality of congestion and timing outcomes. Chapter 2 will provide an in-depth analysis of different mixed-size placers, which are placement algorithms designed to handle multiple types of cells simultaneously during the physical design phase. Specifically, the focus will be on placers that aim to concurrently place both standard cells and predefined macros. This simultaneous placement is a critical challenge in modern VLSI design, as it requires balancing the flexibility of standard cells with the fixed dimensions and constraints of macros, while optimizing for performance, area, and routability.

### 1.4.4 Clock Tree Synthesis

The majority of digital designs operate synchronously, where computation advances as the current values of internal state variables and input variables are supplied to combinational logic networks. These networks subsequently produce outputs and determine the next values of the state variables. A clock signal is essential for ensuring the synchronization of all computations occurring across the chip. This signal can be generated externally or through specialized analog circuitry, such as Phase locked loops (PLL) or Delay locked loops (DLL).

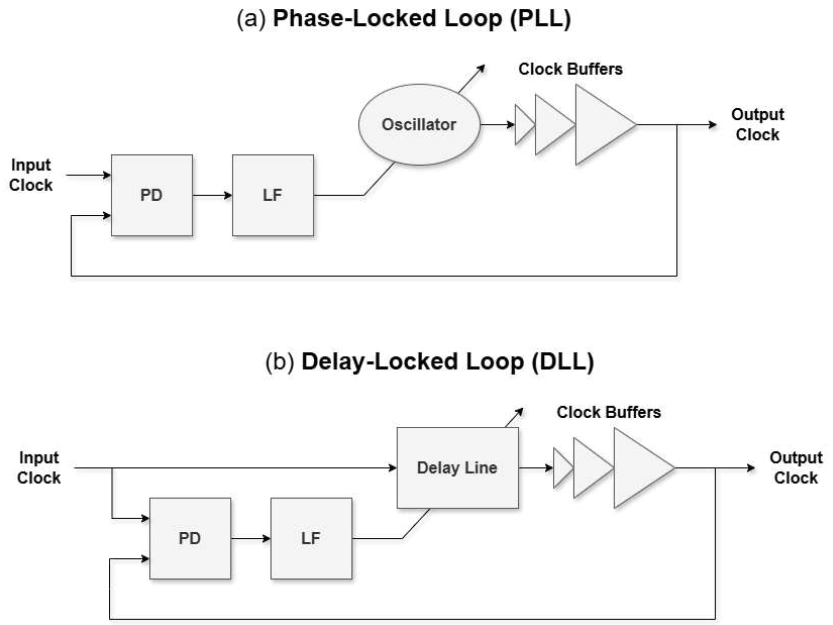



Figure 1.11: Example of (a) PLL and (b) DLL block schemes.

Its frequency may be adjusted through division or multiplication to meet the specific requirements of individual blocks. Once the entry points and sinks for the clock signal, such as flip-flops and latches, are established, clock tree routing is employed to create a clock tree for each clock domain within the circuit. The unique function of the clock signal in synchronizing all computations on the chip distinguishes clock routing from other routing types. The fundamental challenge of clock routing lies in the necessity to deliver the signal from the source to all destinations, or sinks, simultaneously.

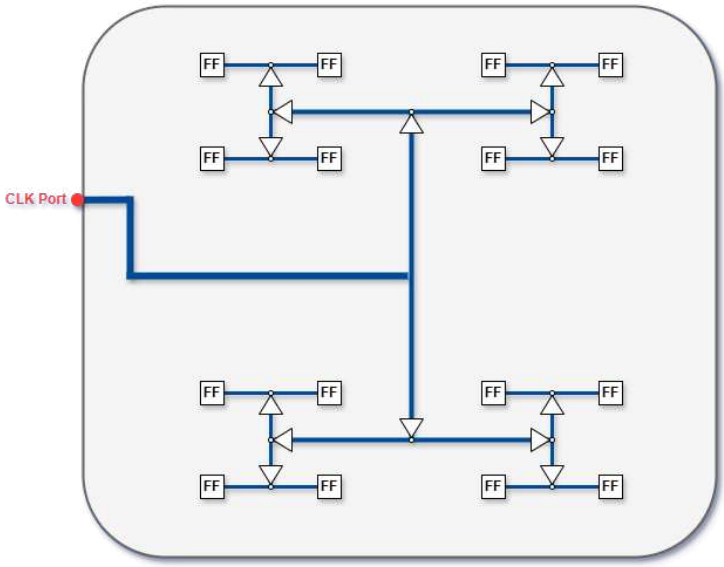



Figure 1.12: Example of H-tree structure.

There are several architectural strategies for constructing clock trees, each with its own trade-offs in terms of skew, power consumption, and implementation complexity. Among the most commonly used structures are H-trees (as seen in Fig. 1.12), X-trees, and balanced binary trees. These topologies are designed to minimize clock skew, that is the difference in arrival times of the clock signal at different sinks, and to ensure uniform delay distribution across the chip.

### 1.4.5 Signal Routing

Routing is a critical stage in the physical design flow of integrated circuits. It involves establishing physical connections between placed standard cells, macros, and I/O pins using metal layers. The goal is to create signal paths that meet design rules, timing constraints, and congestion limits, while minimizing wirelength and delay. Effective routing ensures both functional correctness and performance of the final chip layout. The process of full-chip routing typically consists of three stages: global routing, detailed routing and timing-driven routing (discussed in Sec. 1.4.6.2).

- *Global routing*
  At this stage, the chip layout is abstracted into a coarse routing grid, typically represented as a grid graph. Each edge in this graph corresponds to a routing channel and is associated with a certain routing capacity, which reflects the number of wires that can be routed through that channel without causing congestion. During global routing, the topologies of the nets, the sets of pins that need to be connected, are mapped onto this grid. The goal is to assign each net to a set of routing resources (edges in the graph) in a way that mini-

mizes total wirelength, avoids over-congested regions, and respects design rules and timing constraints. Unlike detailed routing, which defines the exact geometries and layers of the wires, global routing provides a high-level plan that guides the subsequent routing stages. This step is essential for identifying potential routing bottlenecks early in the flow and for enabling congestion-aware optimization in later phases. It also plays a key role in estimating parasitics and timing, which are critical for ensuring that the design meets performance targets.

- *Detailed routing*
  During the detailed routing phase, wire segments are allocated to specific routing tracks. This phase encompasses several intermediate tasks and decisions, including *net ordering*, determining the sequence in which nets should be routed, and *pin ordering*, which dictates the connection sequence of pins within a net. These two factors present significant challenges in sequential routing, where nets are routed individually. The order of nets and pins can significantly influence the quality of the final solution. Detailed routing aims to enhance global routes and generally does not modify the net configurations established during global routing. Therefore, if the global routing outcome is suboptimal, the quality of the detailed routing result will also be adversely affected. To establish net ordering, each net is assigned a numerical importance indicator, referred to as *net weight*. Nets that are timing-critical, connect to multiple pins, or serve specific functions, such as delivering clock signals, may be assigned high priority. It is essential for high-priority nets to minimize unnecessary detours, even if this requires diverting other nets.

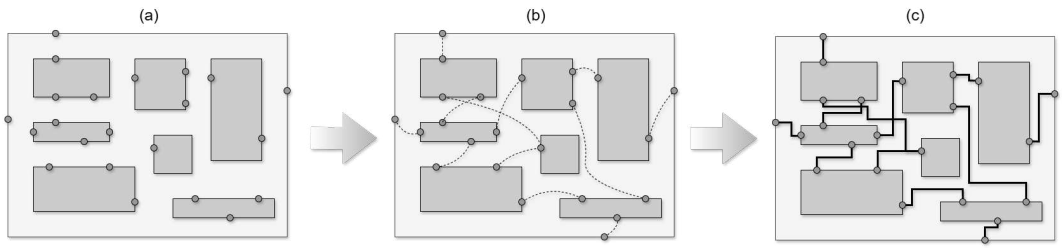



Figure 1.13: Example of: (a) a placement, (b) a global routing, and (c) a detailed routing.

### 1.4.6 Timing Closure

The configuration of an IC must fulfill not only geometric specifications, such as non-overlapping cells and routability, but also adhere to the timing requirements of the design, including *setup* and *hold* constraints. The process of optimization that

addresses these specifications and requirements is commonly referred to as **timing closure**. This process incorporates point optimizations previously discussed, such as placement (Sec. 1.4.3) and routing (Sec. 1.4.5 and Sec. 1.4.4), along with specialized techniques aimed at enhancing circuit performance. The main aspects of timing closure are the following:

- *Timing-driven placement* (Sec. 1.4.6.1) aims to reduce signal delays by strategically assigning locations to circuit components.
- *Timing-driven routing* (Sec. 1.4.6.2) focuses on minimizing signal delays through the selection of routing topologies and specific paths.
- *Physical synthesis* (Sec. 1.4.6.3) enhances timing by modifying the netlist doing changes such as:
  1. Adjusting the sizes of transistors or gates: increasing the width-to-length ratio of transistors to reduce delay or enhance the drive strength of a gate.
  2. Adding buffers to nets to lower propagation delays.
  3. Reorganizing the circuit along its critical paths.

For an extended period, the delay in signal propagation within logic gates was the predominant factor contributing to circuit delay, while the delay caused by wiring was considered minimal. Consequently, the placement of cells and the routing of wires did not significantly influence circuit performance. However, beginning in the mid-1990s, advancements in technology scaling greatly amplified the relative significance of delays induced by wiring, thereby rendering high-quality placement and routing essential for achieving timing closure. Timing optimization engines are required to quickly and accurately estimate circuit delays to enhance circuit timing. Timing optimizers modify propagation delays across circuit components, primarily aiming to meet timing constraints, which include

- *Setup constraints* that dictate the duration a data input signal must remain stable prior to the clock edge for each storage element. Setup constraints ensure that no signal transition occurs too late.

$$T \geq t_{comb} + t_{setup} + t_{skew} \tag{1.1}$$

  T refers to the clock period, $t_{comb}$ denotes the maximum path delay through combinational logic, $t_{setup}$ indicates the setup time of the receiving storage element (such as a flip-flop), and $t_{skew}$ represents the clock skew. To determine if a circuit adheres to setup constraints, it is necessary to estimate the duration of signal transitions as they propagate from one storage element to another. This

delay estimation is generally performed using Static timing analysis (STA), which calculates Actual arrival time (AAT) and Required arrival time (RAT) for the pins of each gate or cell. STA efficiently detects timing violations and identifies their sources by tracing critical paths within the circuit that contribute to these timing issues. Due to efficiency considerations, STA does not take into account the functionality of the circuit or specific signal transitions. Instead, it presumes that every cell transmits every 0-1 (1-0) transition from its inputs to its outputs, with each propagation occurring at the maximum delay. Consequently, the results obtained from STA are frequently overly pessimistic for larger circuits. A crucial measure for a specific timing point is the **timing slack**. This is defined as the difference between RAT and AAT, expressed as

$$Slack = RAT - AAT$$

A positive slack signifies that the timing requirements are satisfied, meaning the signal arrives prior to the necessary time, whereas a negative slack indicates a timing violation, where the signal arrives after the required time.

- *Hold-time constraints* that determine the duration a data input signal must remain stable following the clock edge at each storage element. Hold violations may arise when a signal path is excessively short, enabling a receiving flip-flop to capture the signal in the current cycle rather than in the subsequent cycle.

$$t_{comb} \geq t_{hold} + t_{skew}$$

$t_{comb}$ denotes the maximum path delay through combinational logic, $t_{hold}$ d is the hold time required for the receiving storage element, and $t_{skew}$ represents the clock skew. Since the hold-time constraint is not affected by the clock period, simply reducing the clock frequency does not resolve hold-time violations. For this reason, hold-time constraints are typically addressed after the clock network has been synthesized, when the actual delays introduced by the clock tree are known and can be accurately analyzed.

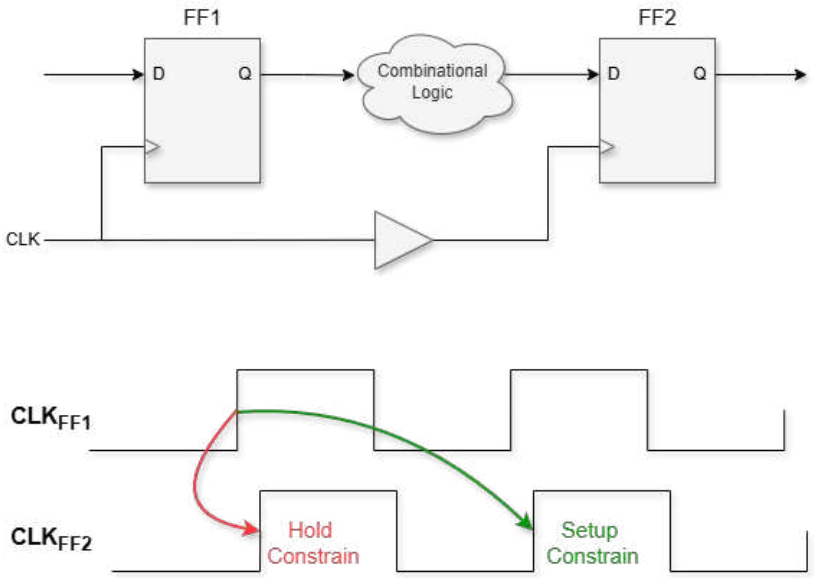



Figure 1.14: Hold and setup constraints.

#### 1.4.6.1 Timing-Driven Placement

Timing-driven placement (TDP) enhances circuit delay to either meet all timing requirements or maximize the clock frequency. It leverages the outcomes of STA to pinpoint critical nets and seeks to reduce signal propagation delay across these nets. Generally, TDP aims to minimize one or both of the following:

1. *Worst negative slack (WNS)*

$$\text{WNS} = \min_{\tau \in T}(\text{slack}(\tau)) \tag{1.2}$$

2. *Total negative slack (TNS)*

$$\text{TNS} = \sum_{\tau \in T,\, \text{slack}(\tau) < 0} \text{slack}(\tau)$$

#### 1.4.6.2 Timing-Driven routing

In contemporary integrated circuits, interconnections significantly impact overall signal delay. Therefore, interconnect delay becomes a critical factor during the routing phases. Timing-driven routing aims to reduce either or both of the following:

1. *Maximum sink delay*, defined as the highest interconnect delay from the source node to any sink of a specific net.

2. *Total wirelength*, which influences the load-dependent delay of the net's driving gate.

#### 1.4.6.3 Physical synthesis

For a chip to function correctly in relation to setup constraints, it is essential that AAT is greater than or equal to RAT at all nodes. If any nodes violate this condition, resulting in negative slack, physical synthesis, a series of timing optimizations, will be implemented until all slacks are non-negative. The optimization process encompasses two key components: *timing budgeting* and *timing correction*. In the timing budgeting phase, target delays are assigned to arcs along timing paths to facilitate timing closure during the placement and routing phases (Sec. 1.4.3 and Sec. 1.4.5), as well as during timing correction. Timing correction involves modifying the netlist to satisfy timing constraints through various operations such as:

- *Gate Sizing*: in the standard-cell design approach, each logic gate is generally available in various sizes that relate to different drive strengths. Drive strength refers to the current that the gate can supply during its switching operation. A larger gate size results in lower output resistance, enabling it to drive a greater load capacitance with reduced load-dependent delay. However, a larger gate size also incurs a higher intrinsic delay due to the parasitic output capacitance inherent to the gate. Resizing transformations adjust the size of critical logic gates to reduce the delay.

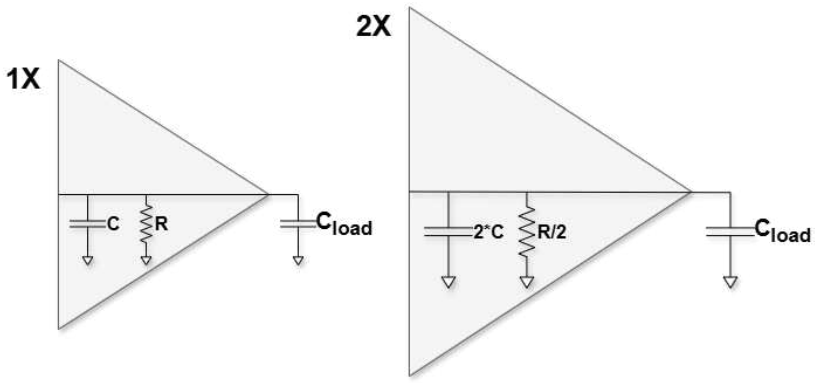



Figure 1.15: Example of the diffrence in internal resistance and capacitance changing the drive strength of a cell.

- *Buffering*: a buffer functions as a gate, usually consisting of two inverters connected in series, which regenerates a signal while preserving its original functionality. Buffers can enhance timing delays by either accelerating the circuit or acting as delay elements, and they can also adjust transition times to enhance signal integrity and mitigate variations in coupling-induced delays. However, a significant disadvantage of buffering techniques is their consumption of available area and the increase in power usage. Despite the careful application of buffering in contemporary tools, the quantity of buffers has been on the rise

in large designs due to trends in technology scaling, where interconnects are becoming slower relative to gates. In modern high-performance designs, buffers may account for 10-20% of all standard cell instances, and in some cases, this figure can reach up to 44% [13].

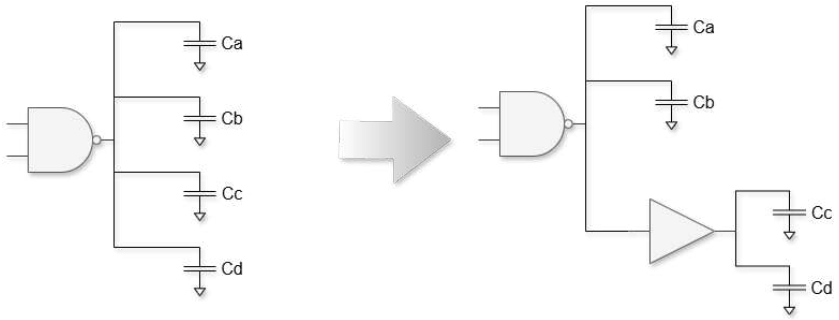



Figure 1.16: In this example the buffer helps to partially shield the load capacitance seen by the NAND gate.

- *Netlist Restructuring*: the netlist can be adjusted to enhance timing performance. These modifications should not affect the circuit's functionality; however, they may involve the addition of extra gates or the reconfiguration of connections between existing gates to strengthen driving capability and improve signal integrity. Some of the possible methods used are:

  1. *Replication*
     The replication of gates can mitigate delay when a gate with considerable fanout experiences slowness due to its fanout capacitance and when a gate's output diverges in two distinct directions, complicating optimal placement for the gate.

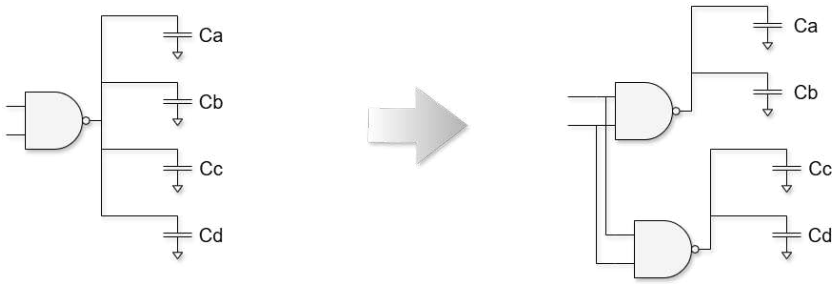



Figure 1.17: Example of replication, the duplicated gate helps to reduce the fanout.

  2. *Redesign of fanin tree*
     The logic design phase typically yields a circuit characterized by the least number of logic levels. Reducing the maximum number of gates along the path connecting sequential elements generally results in a balanced

circuit, ensuring comparable path delays from inputs to outputs. Nevertheless, since input signals can arrive at different times, the circuit with the minimum levels may not achieve optimal timing while a new unbalanced network could have a shorter input/output path.

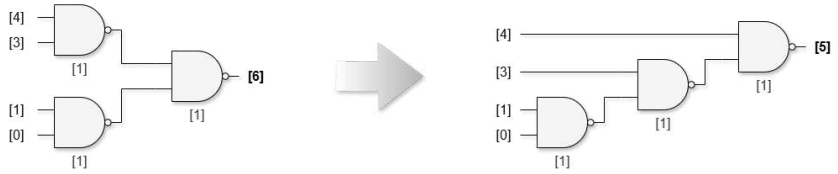



Figure 1.18: Example of fanin tree redesign that allow to achieve a lower delay.

3. *Redesign of fanout tree*
   It's possible to enhance timing by redistributing the output load capacitance within a fanout tree to minimize the delay of the longest path.

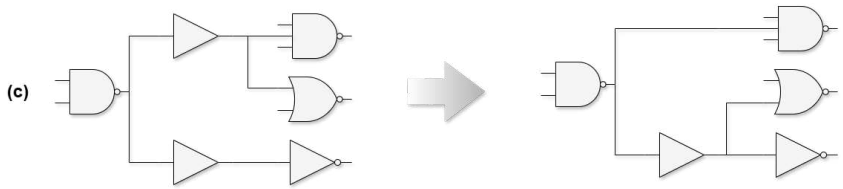



Figure 1.19: Example of fanout tree redesign to reduce the load capacitance of the first path.

4. *Swapping commutative pins*
   While the input pins of a logic gate are logically equivalent, they exhibit different delays to the output pin in the actual transistor network. A general guideline for pin assignment is to allocate a signal that arrives later to an equivalent input pin with a shorter input-output delay, and vice versa.

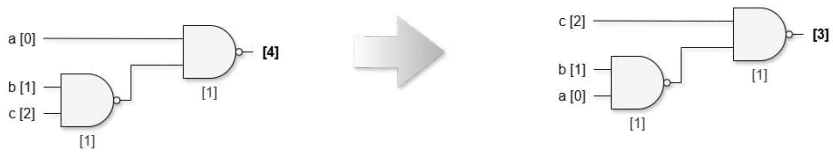



Figure 1.20: Example of swapping commutative pins.

5. *Gate decomposition*
   Gates with multiple inputs typically exhibit increased size and capacitance. Decomposing these multi-input gates into smaller, more efficient gates can reduce both delay and capacitance while preserving the same Boolean functionality.

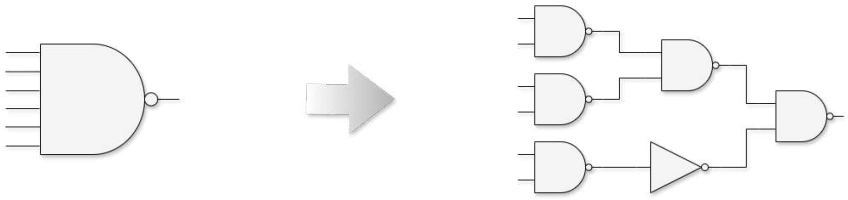


Figure 1.21: Example of gate decomposition.

6. *Boolean restructuring*
   Boolean logic can be realized in various forms within digital circuits and some of them can be more efficient from a timing standpoint.

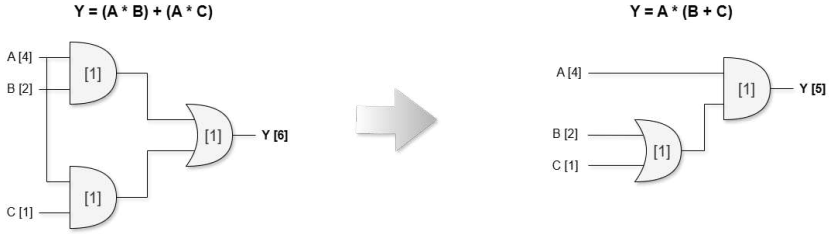



Figure 1.22: Example of Boolean restructuring. Using the distributive law is possible, in this case, to reduce the delay.

7. *Reverse transformations*
   Timing optimizations like buffering, sizing, and cloning can increase area and cause cell overlaps, making the design illegal. To restore legality, either reverse these changes (unbuffer, downsize, merge) or run placement legalization after timing fixes.

Having outlined the complete sequence of steps involved in the physical design flow, from floorplanning through placement, clock tree synthesis, routing, and ultimately timing closure, it is important to recognize how this phase fits into the broader VLSI design process; once physical design is complete and the layout has been finalized, the flow proceeds to design sign-off, which includes checks such as LVS, DRC, and final timing and power verification. These steps ensure that the design is manufacturable and meets all functional and performance specifications before tape-out.

## 1.5 Conclusion

In this chapter, the fundamental principles and stages of the physical design process have been presented, providing the necessary foundation for understanding the more advanced and specialized topics addressed in the remainder of this thesis. Each phase,

from floorplanning to placement, clock tree synthesis, and routing, plays a critical role in shaping the final layout of a VLSI circuit and ensuring that performance, area, and manufacturability constraints are met.

Among these stages, placement represents one of the most computationally intensive and impactful steps, particularly in the context of mixed-size designs, where standard cells must coexist with large macros and pre-designed IP blocks. The presence of these heterogeneous elements introduces significant complexity in terms of optimization, congestion management, and timing closure. To address these challenges, modern placement tools increasingly rely on advanced heuristics and machine learning-based algorithms to explore the vast solution space more effectively.

One notable example is a proprietary mixed-size placer developed by Qualcomm®, which integrates machine learning techniques to improve placement quality and efficiency. This placer has been incorporated into the tool developed as part of this thesis, which is specifically designed to support floorplan exploration. The tool enables designers to experiment with multiple area configurations and macro placements in a flexible and automated manner. By leveraging the internal ML engine, it can generate a wide range of valid macro placements for each area, significantly expanding the design space that a single engineer can explore and analyze.

The next chapter delves into the role and functioning of mixed-size placers, examining their algorithmic foundations and their integration into the floorplan exploration tool. This sets the stage for the experimental analysis presented in Chapter 3, where the tool's capabilities are demonstrated and evaluated in the context of real-world design scenarios.

# Chapter 2

# Mized Size Placers

Placement is a critical phase in the physical design flow of VLSI circuits; its primary objective is to determine the optimal positions of logic components such as standard cells, macro blocks, and memory units within the chip area, while minimizing key design metrics such as total wirelength, congestion, and power consumption. The quality of placement has a direct impact on downstream stages like clock tree synthesis, routing, and timing closure, and it also enables early estimation of physical constraints that can guide earlier design decisions.

In modern integrated circuits, the placement problem becomes significantly more complex due to the mixed-size nature of designs, where small standard cells must coexist with large, pre-designed macros and IP blocks. These heterogeneous elements introduce unique challenges in terms of optimization, spatial constraints, and scalability. Traditional placement algorithms often fall short in effectively handling this complexity, motivating the development of more advanced and specialized mixed-size placers.

This chapter begins by introducing the theoretical foundations of placement, including the basic principles of global placement, wirelength smoothing, density control, and nonlinear optimization. These concepts provide the mathematical and algorithmic basis for understanding how modern placers operate.

Next, the chapter discusses the main challenges in placement, particularly in the context of large-scale, mixed-size designs. This is followed by a classification of placers, distinguishing between analytical, partitioning-based, and stochastic approaches, and a focused discussion on analytical placers, which form the backbone of many state-of-the-art tools.

The chapter then delves into the specifics of mixed-size placement, exploring how the coexistence of macros and standard cells affects placement strategies and tool design. This leads into a comparative overview of three placers: ePlace [1], ePlace-MS [3], and DREAMPlace [2]; highlighting their core algorithms, optimization

techniques, and performance characteristics. Special attention is given to concepts such as eDensity, a density penalty and gradient function based on electrostatic principles and an analogy with deep learning to enable GPU acceleration.

Finally, the chapter presents the Qualcomm® macro placer tool, a proprietary solution that integrates machine learning to enhance macro placement quality. This tool has been incorporated into the floorplan exploration framework developed as part of this thesis, enabling the generation of diverse macro placement alternatives and significantly expanding the design space available to engineers. The practical application and evaluation of this tool will be the focus of Chapter 3.

## 2.1 Theoretical Foundations

Before diving into the practical aspects of mixed-size placement, it is useful to first build a solid understanding of the theoretical principles that underpin modern placement algorithms. This section introduces the key concepts that form the basis of placement strategies, starting from the fundamental goals and constraints of the problem, and moving through the mathematical models used to guide optimization. By exploring topics such as global placement, wirelength smoothing, density control, and nonlinear optimization, we can better appreciate how these techniques contribute to the development of scalable and efficient placers. These foundations will serve as a reference point for the more advanced and specialized approaches discussed in the rest of the chapter.

### 2.1.1 Base Concepts of Placement

At the heart of the placement problem lies the task of determining where each logic component should be positioned within the chip area to achieve optimal performance and manufacturability. This process must account for a variety of constraints, including physical space limitations, connectivity requirements, and timing considerations. To formalize this problem, placement is typically modeled using mathematical abstractions that allow for efficient algorithmic treatment.

A placement instance is formulated as a hyper-graph G = (V,E,R), where V denotes the set of vertices (cells), E denotes the set of hyper-edges (nets) and R denotes the placement region, respectively. A legal solution satisfies the following three requirements:

1. Every cell is accommodated using enough free sites in the placement region.
2. Every cell is horizontally aligned with the boundaries of one placement row.
3. There is no overlap between cells or macros.

Based on the legality constraint, a placer targets minimizing the total HPWL of all the nets. Let v = (x, y) denote a placement solution, where $x = \{x_i | i \in V_m\}$ and $y = \{y_i | i \in V_m\}$ are the x- and y-coordinates of the cells, respectively. The HPWL of each net $e$ is defined as:

$$\text{HPWL}_e(\mathbf{v}) = \max_{i,j \in e} |x_i - x_j| + \max_{i,j \in e} |y_i - y_j| \tag{2.1}$$

The total HPWL is then computed as:

$$HPWL(\mathbf{v}) = \sum_{e \in E} HPWL_e(\mathbf{v}) \tag{2.2}$$

This metric provides a simple yet effective way to evaluate the quality of a placement.

While the fundamental goal of placement is to minimize wirelength while ensuring legality, addressing this objective in large and complex designs requires a more structured and hierarchical approach. This is where global placement becomes essential. By operating at a higher level of abstraction, global placement provides an initial distribution of cells across the chip area, balancing density and connectivity before moving on to more detailed and fine-grained optimization. The next section introduces the key concepts and mathematical formulations that underpin this critical phase of the placement process.

### 2.1.2 Base Concepts of Global Placement

Global placement represents a crucial sub-phase within the overall placement process, where the goal is to determine approximate positions for all cells and macros across the chip area before proceeding to detailed placement. Unlike detailed placement, which focuses on legalizing and refining positions, global placement operates at a coarser level of granularity, aiming to optimize global objectives such as wirelength and cell density distribution.

This phase is typically formulated as a constrained optimization problem, where the placer seeks to minimize a cost function, usually the total wirelength, while satisfying spatial constraints to avoid excessive cell overlap and routing congestion; to facilitate this, the placement region is uniformly partitioned into a grid of rectangular bins,denoted as $B$, which serve as the basic units for estimating and controlling cell density.

Based on a placement solution $v$, let $\rho_b(v)$ denote the density of each grid $b$ as shown in Eq. 2.3.

$$\rho_b(v) = \sum_{i \in V} l_x(b,i)\, l_y(b,i) \tag{2.3}$$

Here, $l_x(b,i)$ and $l_y(b,i)$ denote the horizontal and vertical overlaps between the grid $b$ and the cell $i$. Both $l_x(b,i)$ and $l_y(b,i)$ exhibit a rectangular shape, which is not differentiable at boundary points. As Eq. 2.4 shows, a global placement problem targets a solution $\mathbf{v}$ with minimum total HPWL subject to the constraint that the density $\rho_b(\mathbf{v})$ of all the grids is equal to or below a predetermined target placement density $\rho_t$.

$$\min_{\mathbf{v}} HPWL(\mathbf{v}) \quad \text{s.t.} \quad \rho_b(\mathbf{v}) \leq \rho_t, \quad \forall b \in B \tag{2.4}$$

This formulation ensures that the placement not only minimizes wirelength but also maintains a balanced distribution of cells across the chip, which is essential for routability and timing closure in later stages.

To effectively solve the global placement problem, it is necessary to rely on optimization techniques that can handle both the objective function and the associated constraints in a computationally efficient manner. However, one of the main challenges in this context is the non-differentiability of the wirelength cost function, which complicates the use of gradient-based optimization methods. To address this, various smoothing techniques have been introduced to approximate the wirelength in a differentiable form, enabling faster and more stable convergence during optimization. The next section explores one of the most widely used smoothing models in placement algorithms.

### 2.1.3 Wirelength Smoothing

In global placement, the total wirelength, typically measured using the HPWL, serves as a key objective to minimize. However, the HPWL function is not differentiable. This lack of smoothness poses a significant obstacle for optimization algorithms that rely on gradient information, such as those based on nonlinear programming or machine learning.

To overcome this limitation, several smoothing techniques have been developed to approximate the HPWL with differentiable functions, these approximations allow for the use of efficient gradient-based solvers, improving both the convergence rate and the quality of the final placement. One widely adopted model is the **Weighted-Average (WA)** wirelength model, which provides a smooth approximation of the bounding box of a net by using exponential weighting. The horizontal component of the WA model for a net $e$ is defined as:

$$W_e(\mathbf{v}) = \left( \frac{\sum_{i \in e} x_i \exp(x_i/\gamma)}{\sum_{i \in e} \exp(x_i/\gamma)} - \frac{\sum_{i \in e} x_i \exp(-x_i/\gamma)}{\sum_{i \in e} \exp(-x_i/\gamma)} \right) \tag{2.5}$$

Where $\gamma$ is the smoothing parameter, which can be used to control the modeling accuracy; smaller values of $\gamma$ yield to a closer approximation to the true HPWL, while larger values improve differentiability and numerical stability.

While smoothing techniques address the differentiability of the wirelength objective, they do not resolve another critical aspect of global placement: the need to control cell density across the layout. Without proper density management, cells may cluster in certain regions, leading to routing congestion and timing violations; to tackle this, placement algorithms introduce a density penalty mechanism, which is discussed in the following section.

### 2.1.4 Density Penalty

In addition to minimizing wirelength, a legal and routable placement must ensure that no region of the chip becomes overly congested. This is particularly important in modern IC designs, where the number of bins $|B|$ used to discretize the placement region can be millions. Enforcing individual density constraints for each bin would be computationally infeasible in practice, to address this, all the constraints are typically aggregated into a single density penalty function $N(\mathbf{v})$, as shown in Eq. 2.6. This function penalizes placements that exceed the target density $\rho_t$, and by construction, $N(\mathbf{v}) = 0$ if and only if all density constraints are satisfied.

$$\rho_b(\mathbf{v}) \leq \rho_t, \quad \forall b \in B \iff N(\mathbf{v}) = 0 \tag{2.6}$$

Quadratic placement approaches usually model the density penalty as a linear or quadratic function, which can be easily integrated into their objective function. Nonlinear placers have no constraints on the order of modeling functions thus are able to design the penalty in more flexible ways. In all three discussed papers, the placement instance is modeled as an electrostatic system, where the density penalty function $N(\mathbf{v})$ is interpreted as the system's potential energy, This analogy allows for smooth and continuous modeling of repulsive forces that naturally spread cells across the layout.

Once both the wirelength and density components have been modeled in differentiable form, they can be combined into a unified objective function. This leads to the nonlinear optimization formulation adopted by many modern global placers, as described in the next section.

### 2.1.5 Nonlinear Optimization Formulation

Modern global placement algorithms often rely on nonlinear optimization to simultaneously minimize wirelength and enforce density constraints. By combining the smoothed wirelength function $W(\mathbf{v})$ with the density penalty function $N(\mathbf{v})$, the overall objective function becomes:

$$\min_{\mathbf{v}} f(\mathbf{v}) = W(\mathbf{v}) + \lambda N(\mathbf{v}) \tag{2.7}$$

The penalty factor $\lambda$ is used to control the trade-off between wirelength and density; a higher value of $\lambda$ places more emphasis on satisfying density constraints, potentially at the cost of increased wirelength, while a lower value prioritizes wirelength minimization. Tuning this parameter is essential for achieving high-quality placement results, and many placers adopt adaptive strategies to adjust $\lambda$ dynamically during optimization.

While the mathematical formulation of placement provides a solid foundation for optimization, real-world designs introduce additional layers of complexity. Scalability to millions of components, the coexistence of heterogeneous elements such as macros and standard cells, strict physical constraints, and the need to optimize multiple objectives simultaneously all pose significant challenges. The next section explores these issues in detail, highlighting the practical difficulties that modern placers must overcome.

## 2.2 Challenges in Placement

As integrated circuits continue to grow in complexity and scale, the placement stage of physical design has become increasingly critical and challenging. The task of determining optimal positions for millions of standard cells and macros within a chip layout must account for a wide range of constraints and objectives, all while maintaining computational efficiency; this complexity is further exacerbated by the heterogeneity of modern designs, which often include a mix of small, densely packed standard cells and large, irregularly shaped macros.

Modern placement faces several challenges, including:

- **Scalability**: Designs today may contain millions of cells and thousands of macros.
- **Heterogeneity**: The coexistence of objects with vastly different sizes (standard cells vs. macros) complicates optimization.
- **Physical constraints**: Such as reserved regions, fixed orientations, density targets, and routability requirements.

- **Multi-objective optimization**: Including wirelength minimization, density balancing, timing, and power optimization.

To address these challenges, a wide variety of placement algorithms have been developed over the years, each with different strategies and trade-offs, they can be broadly categorized into three main classes: stochastic, partitioning-based, and analytical approaches. The next section provides an overview of these categories, highlighting their core principles and how they tackle the inherent complexity of the placement problem.

## 2.3 Classification of Placers

Over the years, a wide range of placement algorithms have been proposed to address the increasing complexity of VLSI designs; each approach reflects a different philosophy in tackling the placement problem, balancing trade-offs between solution quality, computational efficiency, and scalability. While the ultimate goal remains the same, finding an optimal or near-optimal placement that satisfies physical and performance constraints, different algorithms adopt distinct strategies to navigate the vast solution space.

Placement algorithms can be broadly categorized into three main classes:

- **Stochastic Placers**: typically rely on simulated annealing techniques (e.g., Timberwolf [14]). Probabilistic acceptance of uphill climbing is employed to help the placer escape from local optima. Although stochastic placement achieves high solution quality, it suffers from significant complexity and a low convergence rate, resulting in limited scalability for large circuits.

- **Partitioning-Based Placers**: systematically reduce the complexity of the problem by dividing the instance (netlist and placement area) into smaller sub-instances (e.g., Capo [15]). Nevertheless, inadequate partitioning during the initial phases may lead to irreversible quality degradation in the final solution.

- **Analytical Placers**: Formulate the problem as a continuous optimization task, often nonlinear, and represent the current state of the art.

Among these categories, analytical placers have gained particular prominence in recent years due to their ability to scale with design complexity and deliver high-quality results. The next section delves deeper into the principles of analytical placement, highlighting why it has become the dominant approach in modern physical design flows.

## 2.4 Analytical Placers

Among the various placement strategies, analytical placers have emerged as the most prominent and widely adopted in modern VLSI design flows. Their strength lies in the ability to model placement as a continuous optimization problem, enabling the use of powerful mathematical techniques to efficiently explore the solution space; unlike stochastic or partitioning-based methods, analytical placers can simultaneously consider global wirelength minimization and local density constraints, making them particularly effective for large-scale and high-performance designs.

Despite their advantages, analytical placers are not without limitations; the optimization problems they solve are often nonlinear and computationally intensive, which can lead to longer runtimes compared to heuristic-based approaches. Nevertheless, their superior solution quality and scalability have made them the foundation of many state-of-the-art placement tools.

Analytical placement techniques can be broadly divided into two main categories: quadratic placement and nonlinear placement.

- **Quadratic Placers**: use quadratic models for wirelength (e.g., FastPlace [16], SimPL [17]). This category of placers approaches the issue by alternating between an unconstrained wirelength minimization phase and a rough legalization (LG) or spreading phase. The wirelength minimization phase typically employs a quadratic wirelength model and aims to minimize the overall wirelength without regard to overlaps among cells. The rough LG phase eliminates overlaps using heuristic methods without explicitly factoring in the wirelength cost. Through the iterative process of these two phases, cells can be progressively spread apart while simultaneously minimizing the wirelength cost.

- **Nonlinear Placers**: addresses the placement challenge using nonlinear optimization methods (e.g., ePlace [1], ePlaceMS [3], RePlAce [18] and DREAMPlace [2]). It establishes a nonlinear optimization problem with a wirelength objective that is subject to a density constraint. By incorporating the density constraint into the objective, gradient descent-based methods can be utilized to seek a high-quality solution.

While analytical placers have demonstrated remarkable performance in standard-cell placement, their effectiveness is further tested in more complex scenarios involving both standard cells and macros. The next section explores the challenges and strategies associated with mixed-size placement, where the coexistence of heterogeneous components introduces new dimensions of complexity to the placement problem.

## 2.5 Mixed-Size Placement

As modern integrated circuits increasingly integrate heterogeneous components, the placement problem evolves from a standard-cell-centric task to a more complex scenario involving both standard cells and large macro blocks. This paradigm, known as *mixed-size placement*, introduces a new set of challenges due to the significant disparity in size, constraints, and behavior between the two object types.

Unlike standard cells, which are small, numerous, and highly regular, macros are large, sparse, and often subject to strict placement and orientation constraints. These differences complicate the optimization landscape and demand more sophisticated placement strategies.

Mixed-size placers must overcome several technical complications, including gradient imbalance during optimization, slow convergence due to the presence of large movable objects, and difficulties in achieving legal placements without sacrificing quality. Despite these challenges, mixed-size placement offers several advantages:

1. *Generality*: A unified treatment of standard cells and macros avoids reliance on ad hoc heuristics.

2. *Scalability*: Advanced techniques such as FFT-based density modeling, preconditioning, and GPU acceleration enable efficient handling of large designs.

3. *Quality*: High-quality solutions can be achieved even in the presence of large macros, with reduced wirelength and congestion.

4. *Extensibility*: The framework can be extended to incorporate additional objectives such as timing, power, and routability.

Historically, mixed-size placement algorithms have been developed following three main paradigms:

- **Two-stage** methods involve two distinct phases: floorplanning and placement. Initially, the location and orientation of macros are established and fixed, followed by placement that focuses on optimizing only standard cells on a global scale. However, the limited data regarding standard cell distribution misleads the floorplanner during the early stages, resulting in a suboptimal floorplan solution that diminishes overall quality.

- **Constructive** (floorplan-guided) approaches leverage the benefits of both floorplan and placement, the floorplanner concurrently optimizes both macros and soft blocks (clusters of standard cells). An incremental placement then distributes standard cells on a local scale, subsequently incremental global and detailed placement further disperses and legalizes the standard cells within

the local scale. Nevertheless, the inherent limitations of partitioning and clustering often lead to suboptimal solutions from a placement perspective. The optimization space for standard cell placement can be significantly reduced, with quality loss that is challenging to recover.

- **One-stage** solutions continue to be favored by most contemporary placement algorithms. By jointly optimizing all components, these methods avoid the limitations of earlier paradigms and offer better integration with modern analytical placement engines.

Several state-of-the-art analytical placers have been extended to support mixed-size placement, each adopting different strategies to address the associated challenges. The next section presents a comparative analysis of three engines, ePlace [1], ePlace-MS [3] and DREAMPlace [2], highlighting their core methodologies, strengths, and limitations.

## 2.6 Comparative Overview: ePlace vs ePlace-MS vs DREAMPlace

In recent years, the evolution of analytical placement has led to the development of several high-performance engines, each pushing the boundaries of scalability, accuracy, and runtime efficiency. Among this there are ePlace [1], ePlace-MS [3] and DREAMPlace [2], three tools that share a common foundation in electrostatics-based modeling but diverge significantly in their implementation strategies, optimization techniques, and hardware acceleration capabilities.

These engines exemplify different stages in the progression of analytical placement: from CPU-based optimization with strong mathematical foundations, to mixed-size support with enhanced preconditioning, and finally to GPU-accelerated frameworks that leverage modern deep learning infrastructures. Understanding their similarities and differences provides valuable insight into the trade-offs involved in designing placement tools for increasingly complex VLSI circuits.

The following table summarizes the core characteristics of each engine, comparing their conceptual models, optimization strategies, and architectural choices:

| | **ePlace** | **ePlace-MS** | **DREAMPlace** |
|---|---|---|---|
| ***Core Concept*** | Electrostatics-based placement using FFT and Nesterov's method | Extension of ePlace for mixed-size circuits | GPU-accelerated analytical placement using deep learning toolkits (PyTorch) |
| ***Density Model*** | Electrostatic analogy with Poisson's equation solved via DCT/DST | Same as ePlace, extended to handle macros with nonlinear preconditioning | Same electrostatic model, implemented with GPU-accelerated DCT/IDCT |
| ***Optimization*** | Nesterov's method with Lipschitz-based step prediction | Nesterov's method with Lipschitz prediction and backtracking. | Multiple solvers via PyTorch (Nesterov, Adam, SGD); automatic differentiation |
| ***Acceleration*** | CPU (OpenMP) | CPU | GPU-based (CUDA kernels for wirelength and density) |
| ***Placement Flow*** | Flat netlist, no clustering | Two-phase: macro legalization (SA) + standard cell refinement | Fully integrated with PyTorch; supports multi-GPU; flat netlist |

Table 2.1: Comparison of ePlace, ePlace-MS, and DREAMPlace

While the comparative overview highlights the architectural and algorithmic distinctions among these tools, a key unifying element lies in their shared use of an electrostatics-inspired density model, known as eDensity. This model plays a central role in guiding the placement process by simulating the physical behavior of charged particles in an electric field. The next section delves into the theoretical foundations and practical implementation of the eDensity model, illustrating how it enables efficient and physically meaningful placement optimization.

### 2.6.1 *eDensity*

A fundamental component shared by the analytical placers ePlace [1], ePlace-MS [3] and DREAMPlace [2] is the electrostatics-inspired density model known as *eDensity*. This model provides a physically intuitive and computationally efficient framework for modeling placement density, by treating the entire placement instance as a two-dimensional electrostatic system where each movable object, a standard cell or a macro, is modeled as a positively charged particle, and the interactions among these particles are governed by classical physical laws such as Coulomb's and Lorentz's.

This analogy transforms the density balancing problem into the search for an electrostatic equilibrium: cells in overfilled regions experience repulsive forces and are pushed toward underutilized areas, while those in less dense regions remain relatively stationary. The result is a natural and smooth redistribution of cells that promotes uniform density and minimizes overlaps, all within a continuous and differentiable optimization framework.

Each node $i$ (a cell or a macro block) in the netlist is transformed to a positively charged particle. The electric quantity $q_i$ of the particle is set to be the node area $A_i$. The motion of a movable cell $i$ is driven by the electric force

$$F_i = q_i \xi_i$$

formulated by Lorentz force law, where $\xi_i$ is the local electric field. Similarly, the cell potential energy $N_i$ is calculated as

$$N_i = q_i \psi_i$$

where $\psi_i$ is the electric potential at cell $i$. By Coulomb's law, the electric field and potential at cell $i$ are the superposition of the contribution from all the remaining cells in the system.

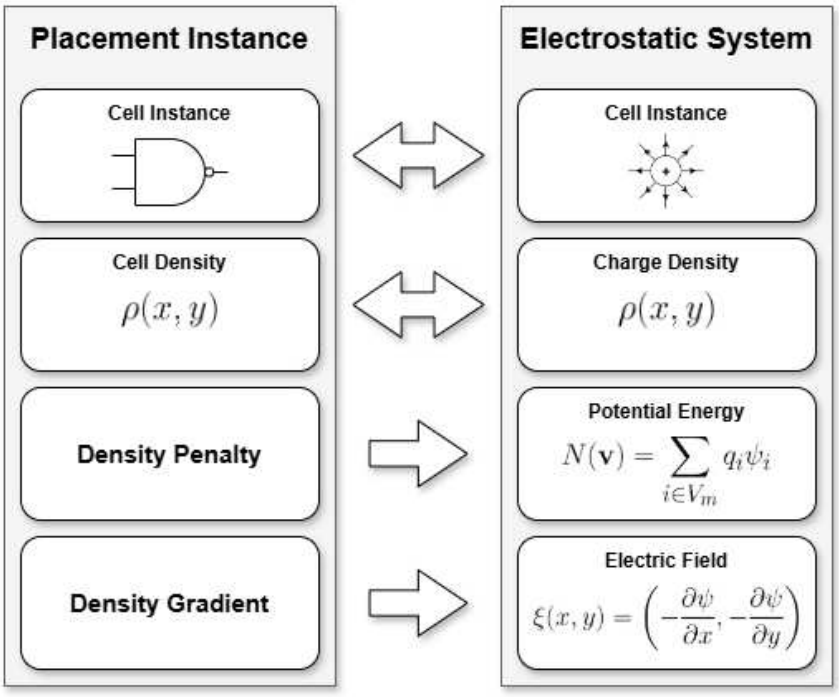
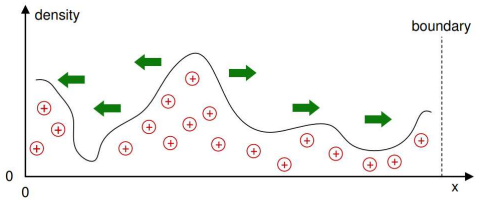
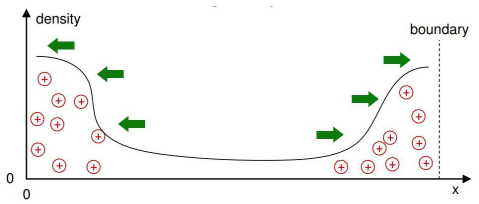



Figure 2.1: Placement instance modeled as an electrostatic system.

An electrostatic system with only positive charges will introduce only repulsion forces. The corresponding equilibrium state would place all cells along the chip boundaries, thereby violating the global placement constraint. To address this, all three considered works remove the DC component (i.e., the zero-frequency component) from the density distribution $\rho(x, y)$, introducing negative charges and ensuring that the integral of the density function over the placement region becomes zero.




(a) Initial Charge Density with DC Component.

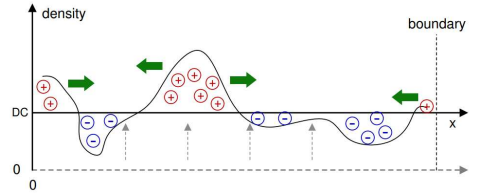



(b) Final Charge Density with DC Component.

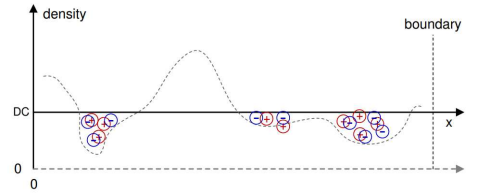
Figure 2.2: Initial and Final Charge Density in Electrostatic Placement with DC component [1].

Since the density function transforms all objects into positive charges, the resulting charge density distribution is initially positive. However, after removing the DC component, under-filled placement regions, those with electric quantity below the original DC level, become negatively charged. Over-filled regions remain positively charged but with reduced electric quantity due to the subtraction of the DC component.

Cells located in positively charged (i.e., highly over-filled) regions are attracted to negatively charged regions, leading to mutual neutralization of positive and negative charges. Meanwhile, cells in negatively charged regions tend to remain stationary. Ultimately, the system reaches an electrostatic equilibrium state characterized by zero net charge density across the entire placement region and a total potential energy reduced to zero.




(a) Initial Charge Density without DC Component.




(b) Final Charge Density after Removing DC Component.

Figure 2.3: Initial and Final Charge Density in Electrostatic Placement after removing DC component [1].

Accordingly, all three papers model the placement density penalty and gradient using the system's potential energy and electric field, respectively.

By Gauss' law, the electric potential distribution $\psi(x, y)$ can be coupled with the density function $\rho(x, y)$ using Poisson's equation. A numerical solution based on spectral methods is proposed in all three considered works to effectively and efficiently solve Poisson's equation. Spectral methods express the solution to a Partial differential equation (PDE) as a summation of basis functions (e.g., sinusoidal and cosine waveforms), with coefficients chosen to satisfy the PDE and its boundary

conditions.

A sinusoidal function, being odd and periodic, naturally satisfies the Neumann boundary condition by diminishing to zero at the boundary of each period. Consequently, sinusoidal wave functions are used as basis functions to represent the electric field. Since the density and potential functions correspond to the derivative and integral of the field function, cosine waveforms are employed as basis functions to express them.

Based on this frequency-domain decomposition (obtained using Discrete Cosine Transform (DCT) and Discrete Sinusoidal Transform (DST), efficiently computed via the Fast Fourier Transform (FFT)), spectral methods are applied to solve Poisson's equation in the placement modeling framework.

The adoption of the *eDensity* model has been instrumental in enabling scalable and high-quality placement, particularly in mixed-size and large-scale designs. However, as circuit complexity continues to grow, there is an increasing need for placement engines that can leverage modern hardware acceleration to further reduce runtime without compromising quality.

This need has led to the development of DREAMPlace [2], which reimagines the placement problem through the lens of deep learning. The next section explores the conceptual analogy between analytical placement and neural network training, and how this analogy enables the use of GPU-accelerated deep learning frameworks to solve placement problems efficiently.

### 2.6.2 Deep Learning Analogy for GPU Acceleration

As the scale and complexity of modern VLSI designs continue to grow, traditional CPU-based placement engines face increasing limitations in terms of runtime and scalability; to address these challenges, recent research has explored the use of GPU acceleration and machine learning frameworks to enhance placement performance. An innovative contribution in this direction is DREAMPlace [2], which reinterprets the analytical placement problem through the lens of deep learning.

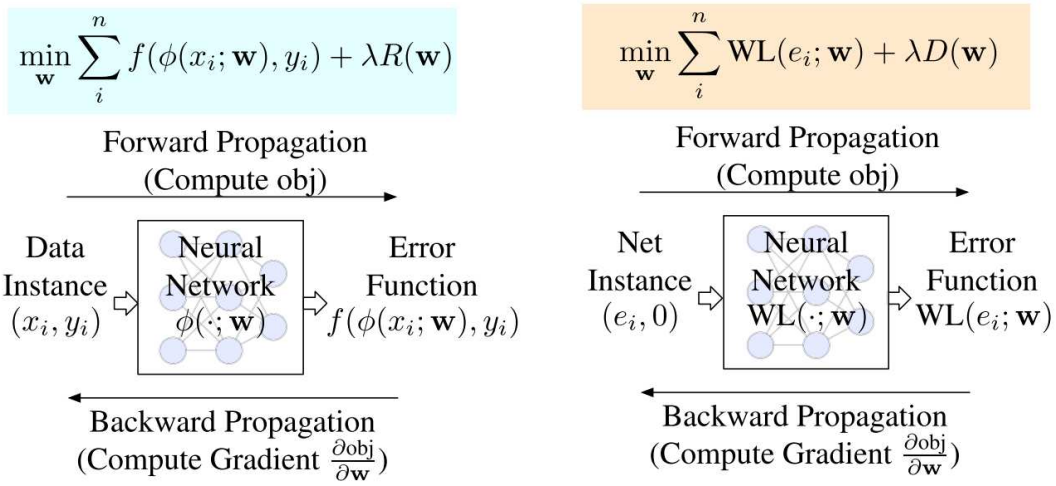
At the core of this approach lies a fascinating analogy: both analytical placement and neural network training can be formulated as large-scale nonlinear optimization problems. This structural similarity enables the reuse of deep learning toolkits, originally developed for training neural networks, to solve placement problems efficiently on GPU architectures. In particular, DREAMPlace [2] leverages PyTorch to implement forward and backward propagation routines, treating cell locations as trainable parameters and using automatic differentiation to compute gradients.

- the wirelength cost in placement corresponds to the prediction error in neural networks,

- the density cost plays a role analogous to the regularization term.

In neural network training, each data instance with a feature vector $\mathbf{x}_i$ and label $y_i$ is processed by the network to produce a prediction $\phi(\mathbf{x}_i; \mathbf{w})$. The training objective is to minimize the total loss over the weights $\mathbf{w}$, which includes both the prediction error and a regularization term $R(\mathbf{w})$ [19].

In the placement analogy, the cell locations $(x, y)$ are collectively represented as $\mathbf{w}$. Each data instance is replaced by a net instance with a feature vector $e_i$ and a target label of zero. The network computes a wirelength cost $WL(e_i; \mathbf{w})$, and using an absolute error function $f(\hat{y}, y) = |\hat{y} - y|$, the total prediction error becomes $\sum_i WL(e_i; \mathbf{w})$. The density cost $D(\mathbf{w})$, independent of the net instances, corresponds to the regularization term.




(a) Train a network for weights. (b) Solve a placement for cell locations.

Figure 2.4: Conceptual analogy between neural network training and analytical placement optimization [2].

This one-to-one mapping between components in analytical placement and neural network training enables the use of deep learning toolkits for placement implementation. Consequently, the placement problem can be solved using a neural network training procedure, involving forward propagation to compute the objective and backward propagation to compute gradients.

This reinterpretation of placement as a differentiable optimization problem has not only enabled significant runtime improvements through GPU acceleration, but also laid the groundwork for hybrid approaches that combine analytical rigor with data-driven intelligence. Building on this foundation, Qualcomm® has developed a proprietary macro placement tool that integrates machine learning techniques and incorporates features from the open-source, GPU-accelerated framework DREAMPlace [2]. The next section introduces this tool, highlighting its unique capabilities

for macro placement exploration and its role in enabling interactive, user-guided design prototyping.

## 2.7 Qualcomm® Macro Placer Tool

In addition to academic research, industrial efforts have also contributed significantly to advancing placement methodologies, particularly in the context of mixed-size and macro-dominated designs. One such contribution comes from Qualcomm®, where I carried out my internship as part of this thesis project; during this experience, I had the opportunity to work directly with a proprietary macro placement tool developed by the company, which integrates machine learning techniques and incorporates features from the open-source, GPU-accelerated placement framework DREAMPlace [2], this tool is designed to enhance the early stages of physical design by enabling rapid exploration of macro placement alternatives, with the goal of improving both design quality and engineering productivity.

Unlike traditional placers that often rely on fixed heuristics or rigid optimization flows, the Qualcomm® tool introduces a more flexible and exploratory approach: it leverages neural networks to generate a diverse set of macro placement candidates, which are then evaluated and ranked based on key physical metrics. This capability is particularly valuable in production environments, where early macro placement decisions can have a profound impact on routability, timing closure, and overall design convergence.

This tool takes as input a set of macro cells and their corresponding placement constraints in the form of two files:

- *LEFlist*: contains a compilation of paths leading to the Library Exchange Format (LEF) files associated with all the cells within the design.
- *Design Exchange Format (DEF)* file: includes details about the design netlist, the coordinates of the HM, the macros, and the locations of the pins.

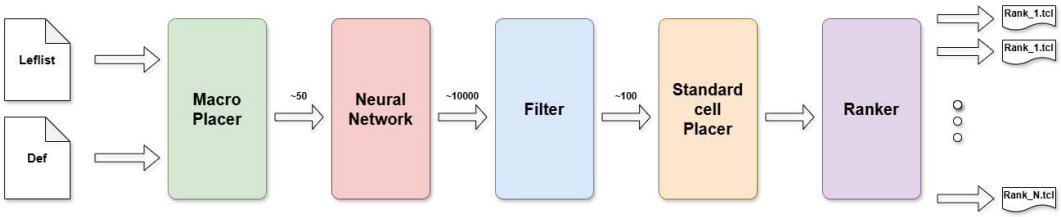



Figure 2.5: Schematic representation of the tool's operations.

In the figure, a schematic representation of the tool's workflow is illustrated. Initially, a macro placer determines the positions of the macrocells using the information

extracted from the input LEF and DEF files. A neural network is then employed to substantially increase the number of possible macro placements, generating a wide variety of layout candidates. This expanded set is subsequently refined through a filtering stage that eliminates unpromising or redundant placements. To simulate realistic design conditions, standard cells are added, enabling accurate estimation of key physical metrics such as congestion and HPWL. Finally, the remaining macro placements are classified based on these metrics to identify the most viable configurations. The outputs of the tool are provided in the form of *.tcl* scripts, which can be sourced and directly used within commercial EDA tools.

To rank the generated macro placements, a Pareto curve is employed; this curve is a standard tool in multi-objective optimization used to evaluate trade-offs between competing objectives and, in this context, the two conflicting goals are wirelength and congestion. The Pareto curve highlights the set of Pareto-optimal solutions, where no placement can be improved in one objective without degrading the other, so the placements that lie on this frontier are considered optimal trade-offs, while those inside the curve are suboptimal, as at least one metric could be improved without negatively affecting the other. This approach allows for a more balanced and insightful evaluation of placement quality.

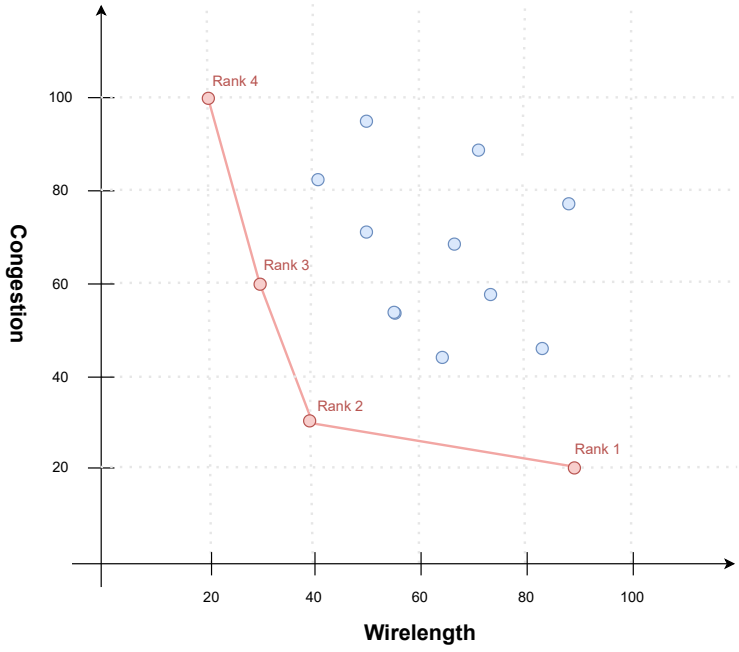



Figure 2.6: Pareto curve illustrating the trade-off between wirelength and congestion.

The macro placer under evaluation has been specifically tailored to produce more

human-like, peripherally driven placements, aiming to mimic the intuition and strategies often employed by experienced physical designers. This design choice is motivated by the observation that such placements tend to enhance routability and lead to better power-performance trade-offs, particularly in designs where ultra-high performance is not the primary objective. Compared to conventional placers, which often prioritize metrics like wirelength above all else, this tool emphasizes a more peripheral distribution of macrocells, aligning with practical design heuristics.

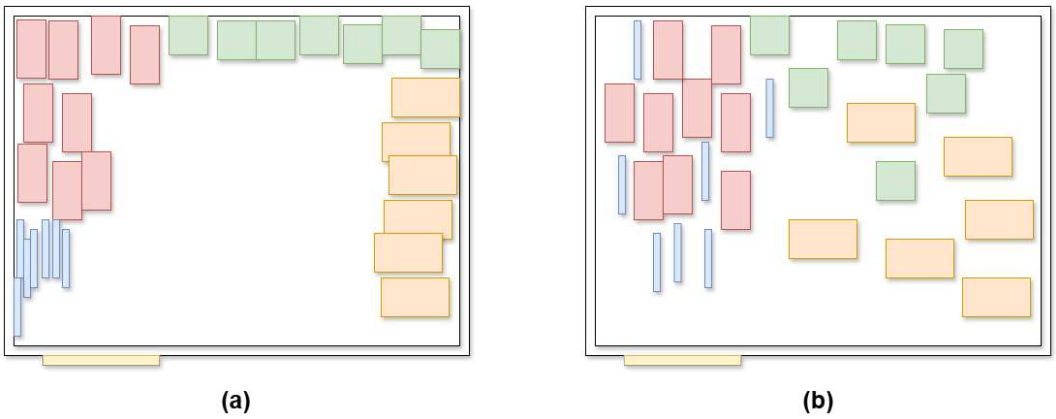



Figure 2.7: Comparison of two macro placements generated using different tools and netlists. One is peripherally driven (a), while the other is focused on wirelength (b).

The Qualcomm® macro placer thus represents a concrete application of the theoretical and algorithmic principles discussed throughout this chapter; by combining analytical placement foundations, GPU acceleration, and machine learning-driven exploration, it exemplifies how modern tools can bridge the gap between academic research and industrial needs. Its ability to generate diverse, human-like macro placements and evaluate them through multi-objective metrics such as wirelength and congestion reflects a broader shift toward more adaptive and designer-aware placement strategies.

The next and final section of this chapter offers a summary of the key insights presented so far, highlighting the evolution of placement techniques, the challenges of mixed-size design, and the emerging role of hybrid, ML-based approaches in shaping the future of physical design automation.

## 2.8 Conclusion

This chapter has provided a comprehensive overview of the theoretical and practical aspects of mixed-size placement in VLSI physical design. It began by introducing the fundamental concepts of placement and global placement, followed by a detailed discussion of wirelength smoothing, density modeling, and the formulation of the

placement problem as a nonlinear optimization task; these foundations set the stage for understanding the complexity of modern placement, which must balance multiple objectives, such as wirelength, congestion, timing, and power, while handling heterogeneous components like standard cells and macros.

The chapter then examined the main classes of placement algorithms, stochastic, partitioning-based, and analytical, highlighting their respective strengths and limitations. Particular attention was given to analytical placers, which represent the current state of the art due to their scalability and ability to integrate multiple objectives into a unified optimization framework. Within this context, we explored the challenges of mixed-size placement and reviewed how modern engines such as ePlace [1], ePlace-MS [3], and DREAMPlace [2] address these challenges using electrostatics-inspired density models and GPU-accelerated computation.

The discussion culminated in the presentation of Qualcomm®'s proprietary macro placer, developed in an industrial setting and designed to support early-stage floorplan exploration through machine learning and designer-aware heuristics; this tool not only exemplifies the practical application of the concepts discussed throughout the chapter but also introduces a more interactive and exploratory approach to macro placement.

Building on these foundations, the next chapter introduces the experimental contribution developed during my internship at Qualcomm®. It presents an enhanced version of the macro placer, extended with a graphical interface and integrated with an area shrink optimization engine. This combined solution enables efficient floorplan exploration and supports early design decisions through a user-guided, multi-objective approach.

# Chapter 3

# Experiment

Building upon the theoretical foundations and placement strategies discussed in the previous chapter, this section presents the experimental contribution developed during my internship at Qualcomm®. The work focuses on extending a proprietary macro placement tool by integrating it with an area shrink optimization engine and a graphical user interface GUI, resulting in a more complete and adaptable solution for floorplan exploration.

This enhanced framework addresses a critical need in physical design: the ability to rapidly explore a wide range of macro placement configurations under varying area constraints, while maintaining control over key physical metrics such as wirelength and congestion. By combining analytical placement principles, GPU acceleration, and machine learning-based exploration, the tool enables designers to evaluate multiple floorplan alternatives early in the design flow, when decisions are most impactful and least costly to revise.

The chapter begins with a detailed description of the GUI and its integration into the existing floorplan flow; particular attention is given to the various configuration options available to the user, which allow for fine-grained control over the exploration process. Following this, a real-world test run is presented to demonstrate the tool's capabilities in an industrial context. In this experiment, a single engineer with limited prior experience was able to generate and analyze eight distinct macro placements across four different area configurations in parallel, showcasing the tool's potential for accelerating early-stage design exploration.

## 3.1 GUI Integration

The enhanced macro placement tool is now accessible through an intuitive and modular graphical user interface GUI, designed to support fast and flexible **floorplan exploration**. As shown in Fig. 3.1, the GUI is organized into three main sections,

each corresponding to a specific stage of the configuration and execution process.

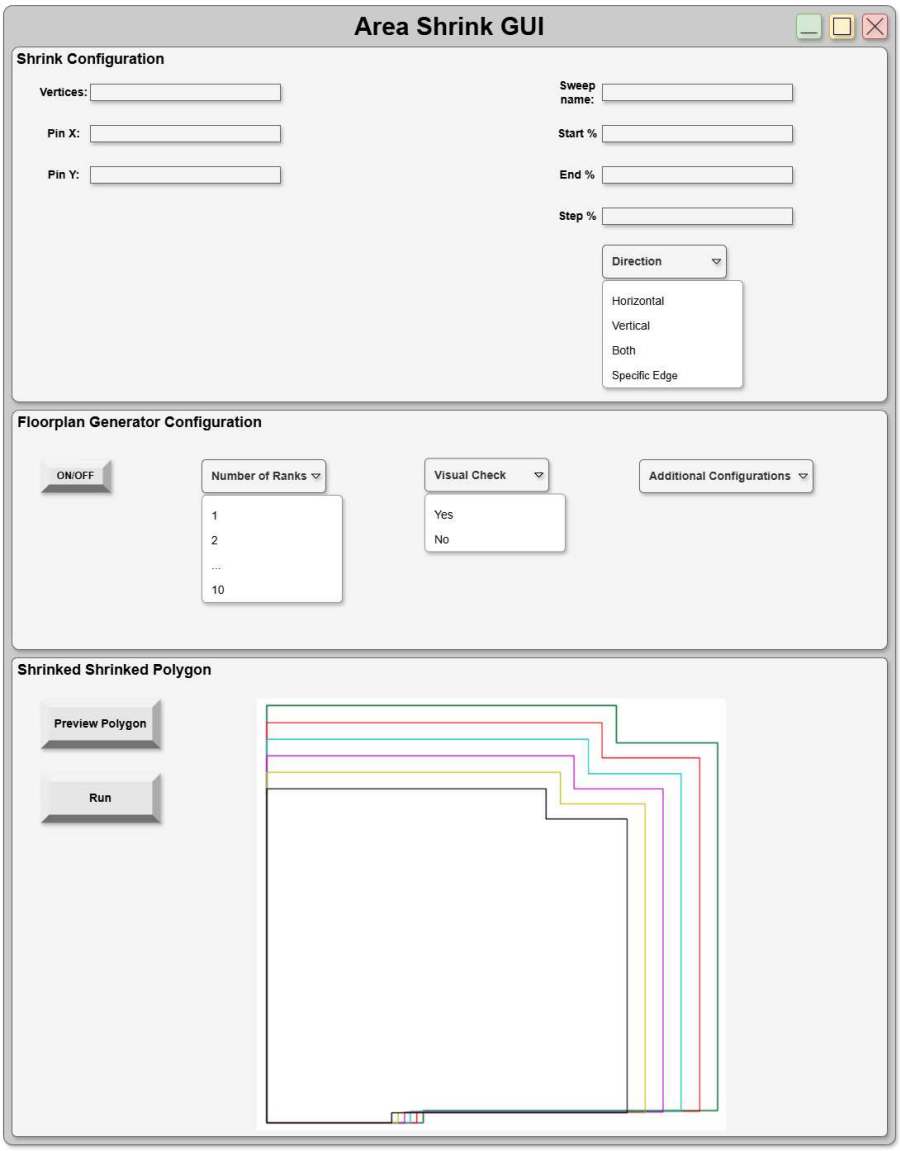



Figure 3.1: The GUI that enables fast floorplan exploration.

The first section, labeled **Shrink Configuration**, allows users to define the geometric and physical context in which the exploration will take place. The process begins by specifying the coordinates of the Hard Macro (HM) to be analyzed, followed by the manual placement of the pins. Once the macro and its interface are defined, the user can configure the area shrink parameters, this is done by specifying the initial and final area percentages (relative to the original macro area) and the step size to be used for the shrink iterations. To maximize flexibility, the GUI offers four shrink direction options: horizontal, vertical, bidirectional, or edge-specific, in this last case, the GUI displays a preview of the macro with all edges numbered, allowing

the user to select a specific edge to shrink. This feature is particularly useful for fine-tuning the floorplan in constrained or asymmetric design contexts.

The second section, named **Floorplan Generator Configuration**, is dedicated to configuring the proprietary macro placement engine. Users can choose whether to activate the engine or bypass it, this is useful in cases where macro placements are already available and the focus is solely on area variation. When enabled, the user can specify the number of ranks to generate for each area configuration; as discussed in Secion 2.7, the macro placer classifies its outputs using a Pareto-based ranking system that balances wirelength and congestion. Additionally, the GUI provides an option to enable a visual check: once the macro placements are generated, a preview window opens automatically for each area, allowing the user to inspect all candidate placements; this interactive step helps identify promising configurations to carry forward into the full physical design flow, while discarding less viable ones early on.

The final section provides a summary of all shrinked macro versions and includes a button to launch the full exploration flow. Once the user initiates the run, the tool automatically generates a structured flow within the industrial EDA environment, as illustrated in Fig. 3.2. The flow is organized as a tree of nodes and branches, where each area configuration corresponds to a parent node responsible for dumping the necessary input files (DEF and LEFlist) and invoking the macro placement engine. For each rank within a given area, a separate branch is created; these branches wait for the macro placement outputs to become available and then automatically source the corresponding *.tcl* scripts, allowing the flow to proceed seamlessly through the standard physical design stages.

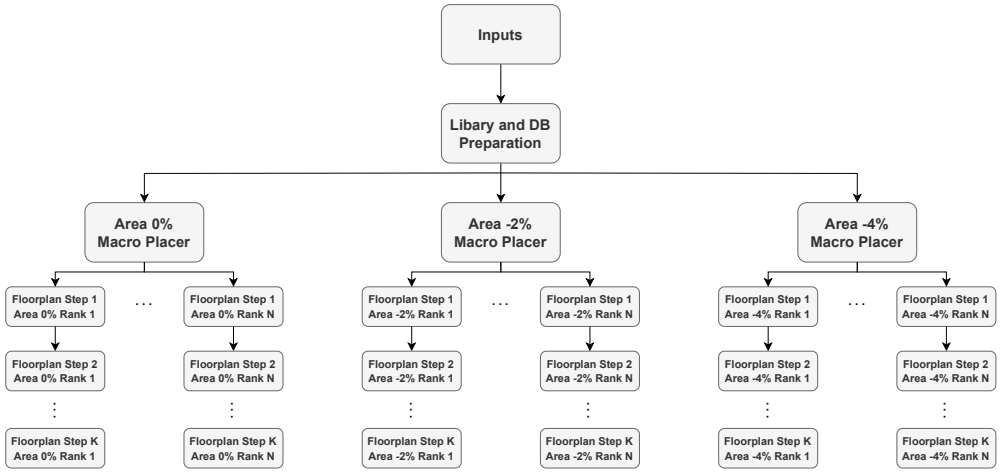



Figure 3.2: Example of flowchart of the Macro Placer process illustrating floorplan exploration under different area reduction scenarios (0%, -2%, -4%), with multiple ranked outputs.

This approach not only simplifies the configuration and execution of complex floorplan exploration tasks, but also ensures that the tool remains accessible and adaptable to a wide range of design scenarios. By abstracting away much of the manual setup typically required in physical design flows, the interface empowers engineers to focus on evaluating design trade-offs and making informed architectural decisions. The seamless integration with commercial EDA environments further enhances its practicality, enabling rapid deployment and efficient iteration within real-world workflows.

To assess the effectiveness of the tool in a realistic industrial context, the next section presents the results of a test run. This experiment demonstrates how the tool can be used to explore multiple macro placement configurations across different area shrink scenarios, highlighting its potential to accelerate early-stage design exploration and support decision-making even in the hands of a single engineer with limited prior experience.

## 3.2 Test Run: Multi-Area Floorplan Exploration

To evaluate the practical utility of the developed tool, a test run was conducted on a real design scenario involving multiple area configurations. The objective was to assess how the tool performs in generating and managing diverse macro placements under varying area constraints, and to demonstrate its effectiveness in supporting early-stage floorplan exploration.

In this experiment, four distinct area configurations were selected, corresponding to area shrink levels of 0%, -1%, -2%, and -3% applied in both horizontal and vertical directions. For each configuration the tool automatically generated multiple macro placements but to streamline the downstream stages of the physical design flow and minimize the need for manual legalization, given that the macro placer is not yet capable of producing fully legalized outputs, two top-ranked placements per area were selected. This resulted in a total of eight macro placements.

It is important to note that the shrink percentage defined in the GUI refers to the total bounding area of the HM. However, since the macro itself remains fixed in size, the effective area available for standard cell placement, referred to as the *placeable core area*, is reduced more significantly.

$$\text{Placeable Core Area Shrink} = \frac{A_{core}^{original} - A_{core}^{shrinked}}{A_{core}^{original}} \cdot 100 \tag{3.1}$$

where $A_{core}^{original}$ is the initial area available for standard cell placement (i.e., total area minus macro area), and $A_{core}^{shrinked}$ is the reduced area after applying the shrink. The following table shows the computed values of *placeable core area shrink* for each

of the four tested configurations:

| **Shrink** | 0% | -1% | -2% | -3% |
|---|---|---|---|---|
| **Placeable Core Area Shrink** | 0% | 3.25% | 7.26% | 10.84% |

Table 3.1: Effect of design shrink on placeable core area reduction.

From the table can be seen that even modest reductions in the total design area result in a significantly larger decrease in the placeable core area. For instance, a global shrink of 2% leads to a 7.26% reduction in the area available for standard cell placement. This highlights the importance of carefully evaluating macro placement strategies under different area constraints, as the available routing and placement resources can be impacted more severely than the shrink percentage might suggest.

To better understand how these area reductions influence macro placement quality and feasibility, the next section presents a visual comparison of the generated macro placements across the different shrink configurations. For each case, the placements are shown both before and after manual legalization, allowing for a qualitative assessment of the tool's output and its adaptability to real-world design constraints.

### 3.2.1 Legalization

While the macro placement engine integrated into the tool is capable of generating high-quality and diverse placement candidates, it currently does not guarantee that the outputs are fully legal (i.e., free from overlaps and aligned to placement grids) and as a result, a manual legalization step is still required to ensure that the generated macro placements can be used in downstream stages of the physical design flow.

This limitation is expected to be addressed in future versions of the tool, in this way once automatic legalization will be supported, the GUI will allow users to choose whether to enable or disable overlap constraints during placement generation. This flexibility will be particularly useful in early design stages, where allowing a limited degree of overlap can lead to more exploratory and unconstrained macro placements. In contrast, during later stages of the flow, users will be able to enforce strict legality to ensure that the outputs are immediately usable for Place & Route (PnR) and signoff.

Figure 3.3 shows the default macro placement used as a baseline in this experiment. This placement was generated using a less recent version of Qualcomm®'s proprietary machine learning-based macro placer. Its origin is evident from certain unconventional placement decisions that would likely not be made by a human designer. For instance, the tall orange macros at the bottom are split into two banks: one placed unusually close to the corner of the layout, and the other more centrally located, this creates a dent between the two banks, a configuration typically avoided

in manual placements due to its potential to cause routability issues. Additionally, the four blue macros in the upper-right region are positioned closer to the center of the core, forming a bump that a human designer would likely smooth out to maintain a more regular and congestion-aware layout. Another clear indication of automated placement is the incomplete preservation of hierarchical grouping: macros belonging to the same logical block or hierarchy are not consistently placed in proximity, which contrasts with common manual floorplanning practices aimed at improving locality and simplifying routing.

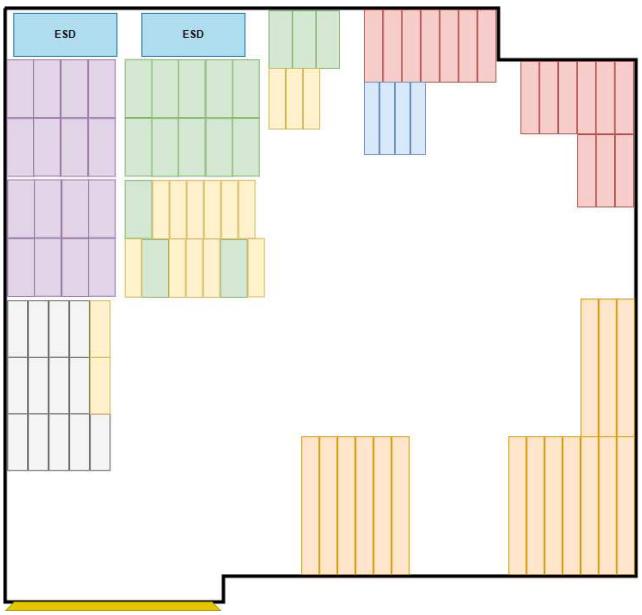



Figure 3.3: Default macro placement.

One notable issue observed in earlier versions was the incorrect handling of fixed cells, such as ESD cells, which are typically required to be placed at specific locations within the design. In the default placement shown above, two ESD cells are correctly positioned in the top-left corner of the layout, this result was achieved by manually adjusting the positions of nearby macros to create sufficient space for the ESD insertion. Since the same adjustment has not yet been applied to the new placements, the ESD cells appear incorrectly placed and overlapping at coordinate (0,0), the bottom-left corner, due to the lack of fixed-cell awareness in the placement engine. Therefore, a similar macro rearrangement will be required for each of the newly generated placements to ensure proper ESD integration.

To address this, a new feature has been added to the GUI that allows users to fix the position of specific cells prior to placement; this includes not only ESD cells, but also other critical components such as Power Mux (PMUX) and other macros with strict placement requirements. This enhancement ensures that future macro placements respect these constraints, improving both the realism and usability of

the generated layouts.

The following figures present the macro placements generated for each area shrink configuration, both before and after manual legalization. These visualizations provide insight into the quality of the raw outputs, the nature of the required adjustments, and the overall effectiveness of the tool in producing viable floorplan candidates under varying design constraints.

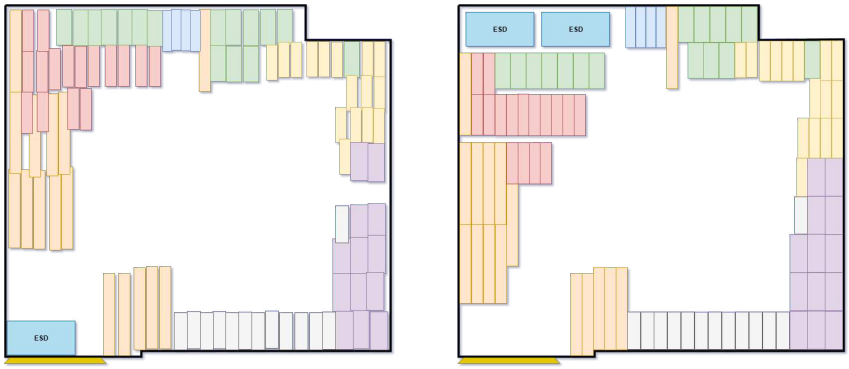



Figure 3.4: Macro Placement Pre and Post legalization: Area 0% / Rank 1.

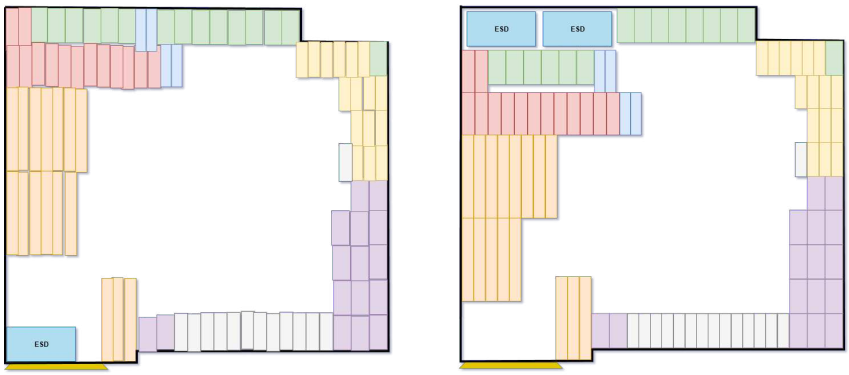



Figure 3.5: Macro Placement Pre and Post legalization: Area 0% / Rank 2.

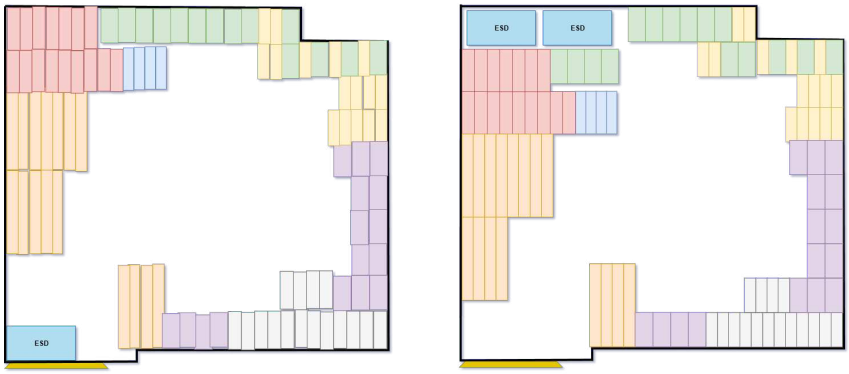



Figure 3.6: Macro Placement Pre and Post legalization: Area -1% / Rank 1.

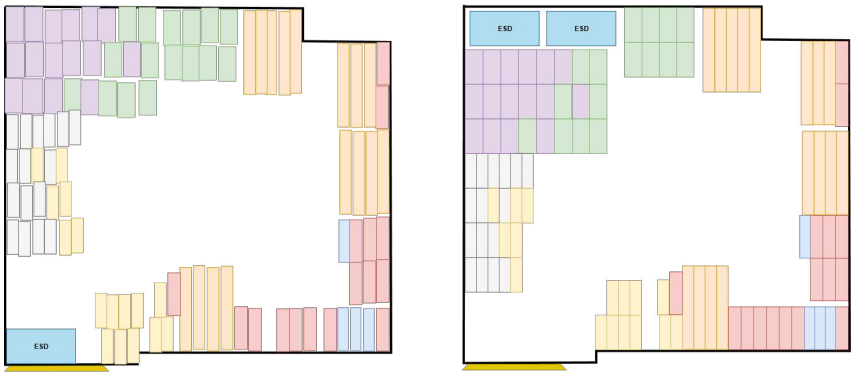



Figure 3.7: Macro Placement Pre and Post legalization: Area -1% / Rank 2.

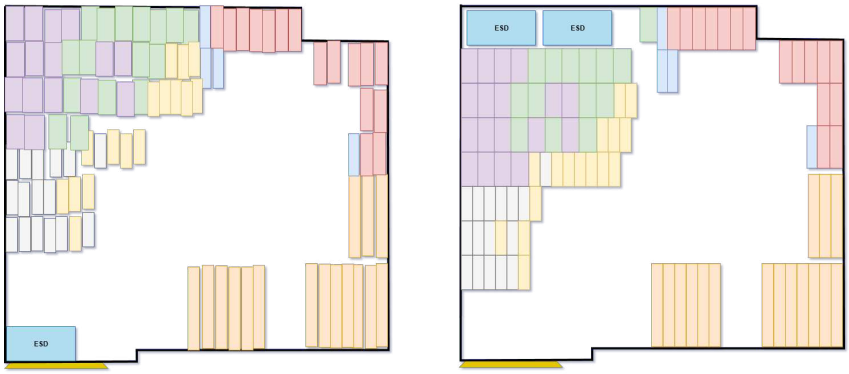



Figure 3.8: Macro Placement Pre and Post legalization: Area -2% / Rank 1.

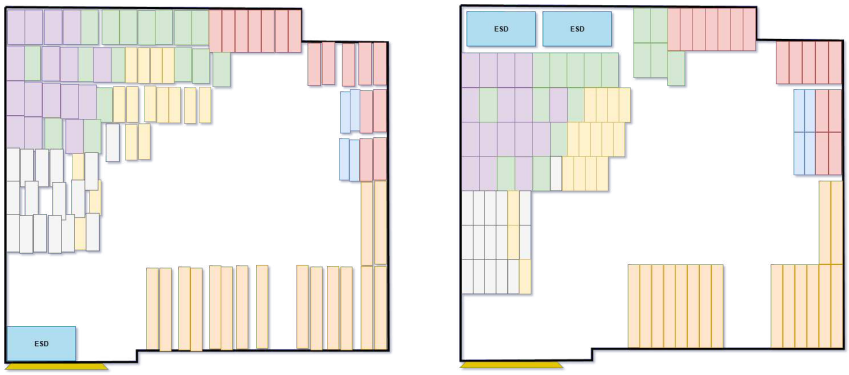



Figure 3.9: Macro Placement Pre and Post legalization: Area -2% / Rank 2.


Figure 3.10: Macro Placement Pre and Post legalization: Area -3% / Rank 1.

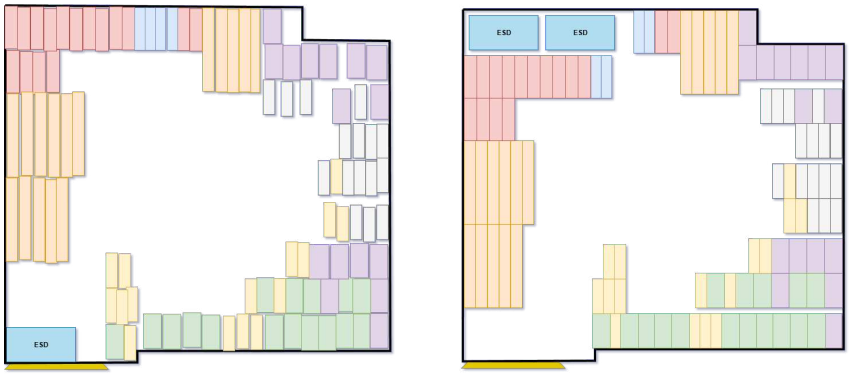



Figure 3.11: Macro Placement Pre and Post legalization: Area -3% / Rank 2.

During the manual legalization process, a deliberate effort was made to minimize modifications to the original macro placements generated by the tool; this choice was motivated by the desire to evaluate the tool's performance as objectively as possible, preserving the spatial intent of the machine learning-based engine and avoiding the introduction of human bias in the layout.

In typical design flows, it is common practice to introduce channels between memory banks to facilitate routing and reduce stacking depth. However, in this experiment, such adjustments were applied only when it was straightforward, on the other hand in several cases, small overlaps between macros were intentionally left unresolved, as eliminating them would have required significant repositioning of the cells, potentially distorting the original output and undermining the purpose of the evaluation.

This conservative approach ensures that the results presented in the following sections reflect the tool's actual capabilities and limitations, providing a more accurate basis for assessing its effectiveness in generating viable macro placements under varying area constraints.

From the visual inspection of the generated macro placements, similarly to what

was observed in the default case, it becomes evident that these layouts were not produced by human designers since several characteristics, easily recognizable across the different configurations, point to the algorithmic nature of the placements.

Notably, the presence of **notches** (irregular indentations in the floorplan), **bumps** (isolated macros protruding from otherwise aligned rows), and the **lack of clear hierarchical grouping** are all signs of a placement strategy driven by optimization objectives rather than design intuition. These features, while potentially acceptable from a purely physical standpoint, often conflict with common human design practices that prioritize symmetry, alignment, and logical grouping for readability, maintainability, and routability.

Such patterns reinforce the idea that the tool operates with a different set of priorities compared to a human designer, focusing on global metrics like wirelength and congestion rather than visual or structural regularity; this distinction is important when interpreting the results, as it highlights both the strengths and the current limitations of machine-generated macro placements in real-world design contexts.

While the visual characteristics of the generated macro placements offer valuable insights into the algorithmic nature of the tool's output, a comprehensive evaluation requires a quantitative analysis of the design metrics that ultimately determine the viability of a physical implementation. To this end, the following section presents the QoR obtained after completing the placement stage for each configuration; metrics such as total wirelength, congestion, timing slack, and utilization serve as objective indicators of the tool's effectiveness in producing layouts that are not only physically realizable but also competitive in terms of performance and efficiency. By comparing these results across different area constraints and placement strategies, we aim to assess the practical impact of the tool's decisions and identify potential areas for improvement in future iterations.

### 3.2.2 Quality of Results at Placement stage

Following the qualitative assessment of the macro placements, this section presents a detailed quantitative evaluation aimed at measuring the practical implications of the generated layouts. The analysis focuses on the QoR obtained immediately after the placement stage, offering a comprehensive view of how each configuration performs in terms of physical feasibility, routing complexity, and timing robustness.

To structure the evaluation, three main categories of metrics are considered: cell metrics, wire metrics, and power metrics; these dimensions capture different aspects of the design's physical characteristics and are visualized through a series of heatmaps. The **cell metrics** heatmap highlights areas of high cell density and potential congestion, which are critical for understanding placement compactness and

routability. The **wire metrics** heatmap provides insight into wirelength distribution and routing pressure, helping to identify regions where excessive interconnect complexity may arise. Finally, the **power metrics** heatmap reveals the spatial distribution of dynamic and leakage power, which is essential for early-stage thermal and power integrity analysis.

In addition to these physical indicators, the section also includes a set of plots illustrating the **setup timing margins** across multiple Process-Voltage-Temperature (PVT) corners. These results are particularly important for evaluating the timing resilience of each placement under realistic operating conditions. By analyzing the slack distribution across corners such as typical-typical (TT), slow-slow (SS), and fast-fast (FF), we can assess how well the placement supports timing closure and identify configurations that may introduce critical path violations or require further optimization in later stages.

Together, these visual and numerical analyses provide a multi-faceted perspective on the quality of the placement solutions. They not only highlight the strengths and limitations of the machine-generated layouts but also serve as a foundation for understanding how early placement decisions propagate through the design flow, ultimately affecting PPA outcomes.

### Cell Metrics Analysis

The heatmap in Fig. 3.12 presents a comparative overview of key cell-related metrics across different area constraint scenarios; these metrics include buffer and inverter area and count, sequential and standard cell area, total standard cell count, and overall cell utilization. All values are normalized with respect to the default configuration, allowing for a direct comparison of how each metric evolves as the available placement area is progressively reduced.

A first observation concerns the stability of the metrics associated with buffers, inverters, and standard cells. Across all configurations, the values for Buff_area, Buff_count, Inv_area, and Inv_count remain consistently close to 1, this indicates that the tool's placement strategy does not significantly alter the number or area of these fundamental components, regardless of the area constraints. Similarly, the total standard cell count and area (Stdcell_count, Stdcell_area) exhibit only minor fluctuations, suggesting that the synthesis and placement stages maintain a comparable logic footprint across all scenarios.

The most notable variations are observed in the Utilization metric; in the configurations where the total area is equal to that of the default case (0% reduction, rank 1 and 2), utilization is slightly lower, this may be attributed to differences in macro organization introduced by the tool, which can affect the distribution and compactness of standard cells. Interestingly, in the case of a 1% total area reduction,

corresponding to a 3.25% decrease in placeable area as reported in Table 3.1, the utilization remains nearly identical to the default, indicating that the tool is capable of adapting to moderate area constraints without compromising placement density.

As expected, in the more aggressive scenarios with 2% and 3% area reductions (corresponding to 7.26% and 10.84% reductions in placeable area, respectively), utilization increases progressively. This trend reflects the natural consequence of fitting the same logic content into a smaller footprint, leading to higher cell density and potentially greater routing complexity in subsequent stages.

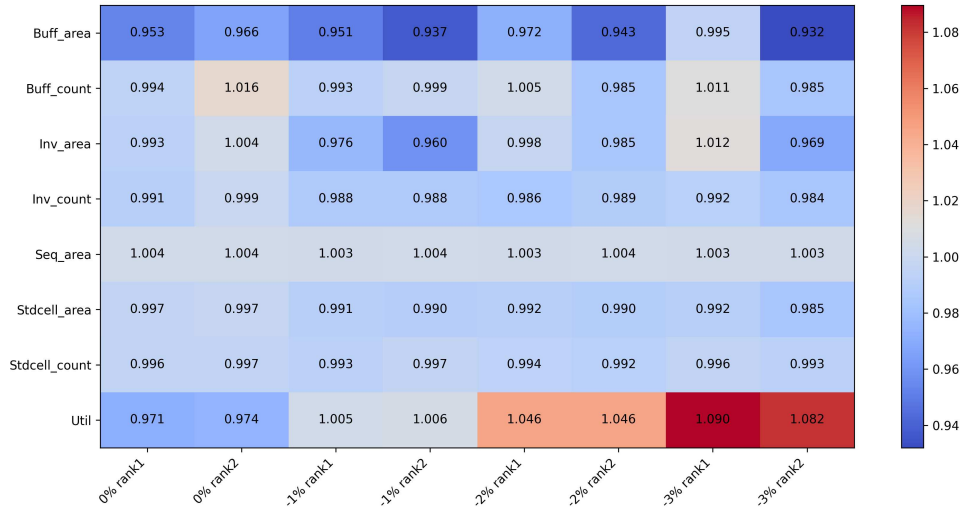



Figure 3.12: HeatMap QoR Placement Cell Metrics.

Overall, this analysis confirms that while the tool maintains consistent behavior in terms of cell instantiation, the utilization metric is sensitive to area constraints and provides an early indicator of the pressure imposed on the layout by tighter design envelopes.

### Wire Metrics Analysis

The heatmap in Fig. 3.13 illustrates the behavior of key wire-related metrics across different area constraint scenarios; The metrics considered include the total number of nets (Net_count), the estimated total wirelength (Wirelength), and the routing overflow (Overflow). All values are normalized with respect to the default configuration, enabling a direct comparison of how these parameters evolve as the available placement area is progressively reduced.

As shown in the heatmap, the Net count remains remarkably stable across all configurations, with values consistently close to 1, this confirms that the logical connectivity of the design is preserved regardless of the macro placement strategy or area constraints, as expected in a flow where synthesis is held constant.

The Wirelength metric also remains close to 1 in all scenarios, but a slight downward trend can be observed as the area decreases. This behavior is consistent with expectations: as the total layout area shrinks, the average distance between macros and standard cells is reduced, leading to a modest decrease in the total interconnect length required to maintain connectivity.

The most significant variations are observed in the Overflow metric, which serves as an early indicator of routing congestion. In the configurations with 0%, 1%, and 2% area reductions, overflow values remain close to or below the baseline, with several cases showing improvements of up to 20% compared to the default. This suggests that the tool is capable of producing placements that are not only compact but also more routable under moderate area constraints.

However, this trend is sharply reversed in the most aggressive scenario, with a 3% area reduction. In this case, the overflow metric increases dramatically, approximately tripling relative to the default configuration. This abrupt degradation indicates that the design has reached a critical threshold where the reduced placement area can no longer accommodate the routing demand without significant congestion, potentially leading to violations or the need for costly design iterations in later stages.

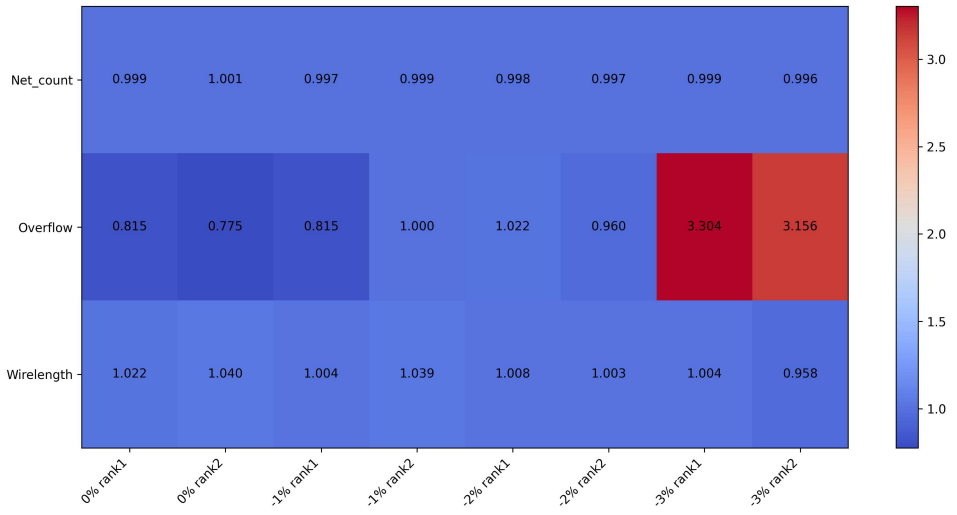



Figure 3.13: HeatMap QoR Placement Wire Metrics.

In summary, while the wire metrics remain largely stable under moderate area reductions, the overflow behavior highlights a clear limit beyond which the placement quality, and consequently the routability, deteriorates rapidly. This insight is crucial for defining practical area constraints in future design iterations.

### Power Metrics Analysis

Figure 3.14 presents the normalized power metrics across various area constraint scenarios, including internal power, switching power, leakage power, and total power. These metrics provide insight into the energy profile of the design immediately after placement, allowing for an early evaluation of power integrity and efficiency.

As shown in the heatmap, the values for internal power, switching power, and total power remain remarkably stable across all configurations. These metrics consistently hover around a normalized value of 1, indicating that the dynamic power components, driven by capacitive loading and switching activity, are largely unaffected by the changes in macro placement and area constraints; this stability suggests that the tool's placement decisions do not introduce significant variations in logic activity or cell selection that would impact dynamic power.

In contrast, the leakage power metric exhibits a modest but consistent increase across several configurations, with values exceeding the baseline by a few percentage points. This behavior may be attributed to the increased cell density and tighter packing in more compact layouts, which can lead to higher leakage due to proximity effects, increased gate count, or shifts in threshold voltage distributions. Although the variation is relatively small, it highlights the sensitivity of static power to layout conditions and reinforces the importance of monitoring leakage even in early design stages.

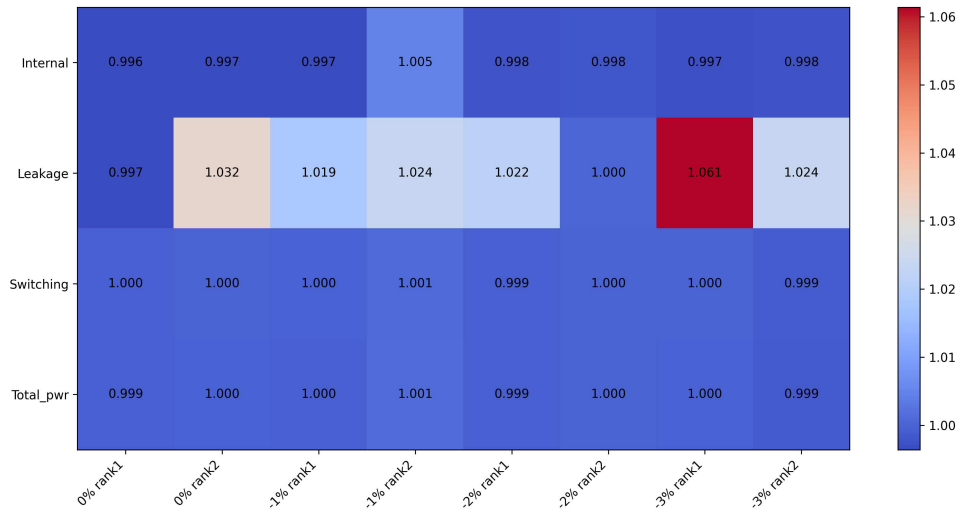



Figure 3.14: HeatMap QoR Placement Power Metrics.

Overall, the power metrics confirm that the tool maintains a stable dynamic power profile across all scenarios, while the observed increase in leakage power under tighter area constraints suggests a potential trade-off between compactness and static power efficiency.

In conclusion, the heatmap-based evaluation of cell, wire, and power metrics provides a comprehensive overview of the physical characteristics of the design immediately after placement. The analysis confirms that the tool maintains consistent behavior across most configurations, with buffer, inverter, and standard cell metrics remaining stable and comparable to the default case. Wire-related metrics show a slight reduction in wirelength with decreasing area, as expected, while routing overflow remains well-controlled up to moderate area reductions, but increases sharply in the most compact scenario. Power metrics reveal stable internal and switching power across all configurations, with a modest increase in leakage power under tighter area constraints.

These results suggest that the machine learning-based placement tool is capable of generating physically viable layouts even under constrained conditions, with predictable trade-offs in density, routability, and power dissipation. However, physical metrics alone do not fully capture the viability of a design. To assess whether these placements support timing closure and meet performance requirements, it is essential to analyze the timing margins across a range of operating conditions.

In the following section, we shift our focus to the timing analysis, examining the behavior of each configuration across multiple PVT corners. Metrics such as WNS, TNS, and the number of failing points will be used to evaluate the timing robustness of the placements and identify configurations that may require further optimization.

### Timing Margin Analysis Across Corners

Figure 3.15 presents the Setup WNS measured across five distinct PVT corners. Due to Company Confidential Information (CCI), the specific definitions of these corners are omitted; however, based on their relative operating conditions, qualitative classifications can be inferred.

*Corner 1*, characterized by high-speed process, voltage, and temperature conditions, can be classified as a FF corner. In this scenario, all configurations exhibit positive slack, indicating successful timing closure. The worst-performing runs are those with -1% area reduction (rank 1) and -2% area reduction (rank 2), which show slightly lower slack values. Interestingly, the best-performing configurations in this corner are -1% rank 2 and -3% rank 2, which outperform even the default placement, despite the reduced area.

*Corner 2*, representing a Slow corner, also shows positive slack across all configurations, though the values are generally lower than those observed in corner 1. The default configuration maintains the highest slack, with the two alternative placements at the same area level showing slightly reduced but comparable performance whie all the other configurations with reduced area perform better then the ones with the original area suggesting that certain compact placements may offer timing

advantages under fast operating conditions.

*Corners 3 and 4*, both classified as SS corners, present a different behavior with respect to the previous ones. In these scenarios, all configurations—including the default—exhibit zero slack, indicating that the design is operating at the edge of timing closure, the only exception is the -3% area reduction (rank 1) configuration, which shows a small but negative slack, suggesting a timing violation introduced by aggressive compaction.

*Corner 5*, which can be considered an extra SS corner, is the most critical in terms of timing. Here, most configurations exhibit negative slack, although the violations are generally small. Once again, the -3% rank 1 configuration stands out as the worst-performing case, with a significantly more negative slack compared to the others.

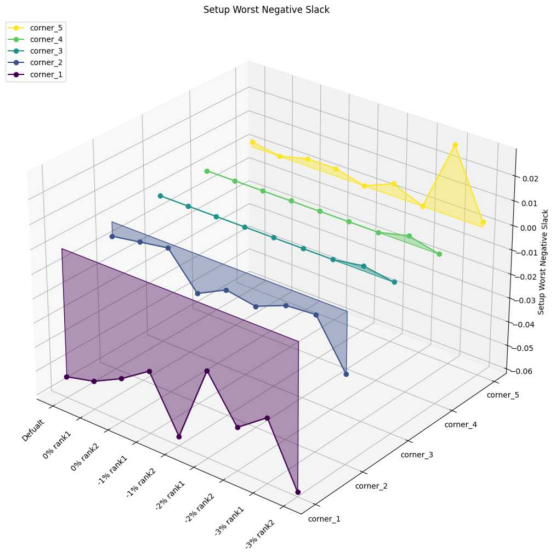



Figure 3.15: Timing QoR Placement Setup Worst Negative Slack.

In summary, the WNS analysis reveals that while most configurations remain timing-clean under typical and fast corners, aggressive area reductions can lead to violations in slower corners. The results underscore the importance of evaluating placement strategies across a diverse set of PVT conditions to ensure robust timing behavior.

In addition to the WNS, Fig. 3.16 reports the TNS across the same five PVT corners. This metric captures the cumulative sum of all negative slack values in the design, offering a broader view of timing robustness by accounting for the number and severity of failing paths.

As anticipated from the WNS analysis, the first four corners, *corner 1* (FF),

*corner 2* (Slow), and *corners 3 and 4* (SS), exhibit zero TNS across all placement configurations. This confirms that, in these scenarios, the designs either meet timing entirely or have only isolated violations that do not accumulate across multiple paths.

The situation changes in *corner 5*, identified as the most critical (extra SS), where several configurations begin to accumulate negative slack, indicating the presence of multiple failing paths. Among them, the -3% area reduction (rank 2) configuration stands out as the worst-performing case, with the highest TNS value. This result is consistent with the WNS analysis and reinforces the conclusion that aggressive area compaction can significantly degrade timing under extreme operating conditions.

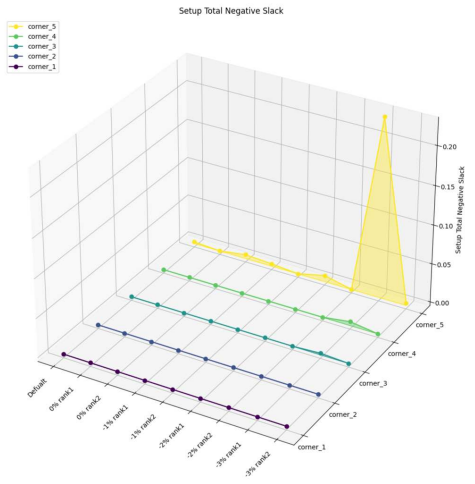



Figure 3.16: Timing QoR Placement Setup Total Negative Slack.

These findings highlight the importance of evaluating not only the worst-case path but also the overall timing distribution, especially when pushing the design toward tighter area constraints.

Finally, Fig. 3.17 reports the number of failing points, i.e., the number of end-points in the design that violate setup timing constraints; this metric complements the WNS and TNS analyses by quantifying the extent of timing violations in terms of affected paths.

As expected, the trend closely mirrors that observed in the TNS plot. In the less critical corners no negative slack is observed, and consequently, the number of failing points remains zero across all configurations. Similarly, in corners 3 and 4, the number of failing points is also zero, except for the -3% area reduction configurations, which introduce a small number of violations.

The most critical behavior is again observed in corner 5 (extra SS). Here, several configurations exhibit non-zero failing points, with the -3% rank 1 configuration standing out as the worst-performing case. This result is consistent with both the

WNS and TNS analyses, confirming that this specific macro placement leads to widespread timing violations under the most pessimistic operating conditions.

In contrast, the remaining configurations in corner 5 maintain a number of failing points comparable to the default case, demonstrating that moderate area reductions can still preserve timing integrity.

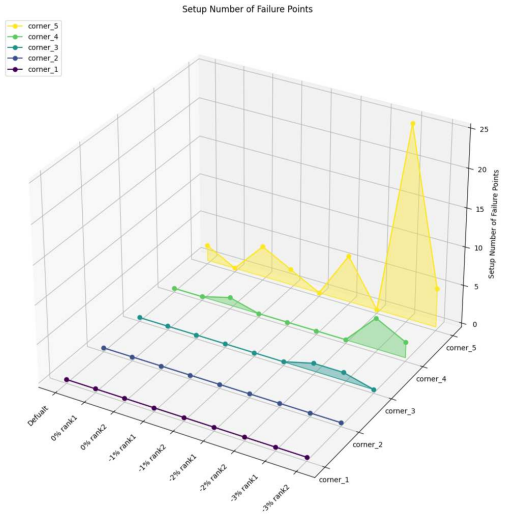



Figure 3.17: Timing QoR Placement Setup Number Failure Point.

In conclusion, the timing analysis across multiple PVT corners reveals a consistent pattern: while most configurations maintain robust timing margins under typical and moderately constrained conditions, aggressive area reductions, particularly the -3% rank 2 configuration, tend to introduce violations in the most pessimistic corners. The WNS, TNS, and number of failing points all confirm that corner 5, classified as extra Slow-Slow, is the most timing-critical scenario. Nevertheless, several configurations, including some with reduced area, demonstrate timing behavior comparable to or even better than the default, highlighting the potential of machine-generated placements to balance compactness and performance when carefully selected.

These results provide a comprehensive view of the placement stage's impact on timing closure and establish a solid baseline for evaluating the downstream effects of routing. In the following section, we extend this analysis by examining the QoR obtained after the routing stage, where additional physical effects such as wire resistance, coupling capacitance, and detailed congestion come into play. This will allow us to assess how well the placement strategies translate into fully implementable designs and whether the observed trends persist or evolve in the final layout.

### 3.2.3 Quality of Results at Post Route Stage

This section presents the QoR obtained after the routing stage, where additional physical effects such as wire resistance, coupling capacitance, and detailed congestion are taken into account. These factors can influence both the physical and timing characteristics of the design, making this stage critical for validating the viability of the placement strategies.

It is important to note that the run corresponding to the -1% area reduction rank 1 **did not converge** during the routing process and this may be attributed to a combination of increased placement density and suboptimal macro organization, which likely led to excessive routing congestion or violations that the tool was unable to resolve within the allowed iteration or runtime limits. As a result, this configuration is excluded from the post-route heatmaps and from the timing plots related to setup and hold analysis.

#### Cell Metrics Analysis

Figure 3.18 shows the post-route cell metrics for the various placement configurations. Overall, the values are very similar to those observed after placement (Fig. 3.12), confirming the stability of the cell-level characteristics throughout the flow. However, some subtle differences can be observed:

- *Buf_ area* values are slightly higher than in the placement stage.
- *Buf_ count* is generally lower than in the placement stage for all configurations, except for -3% rank 2, which shows a slight increase.
- *Inv_ area* and *Inv_ count* remain virtually unchanged.
- *Seq_ area* is identical to the placement values.
- *Stdcell_ area* and *Stdcell_ count* are very close to the previous stage, with negligible variation.
- *Utilization* values are slightly lower (by a few percentage points) in most configurations, except for the two -3% area runs, where a small increase is observed.

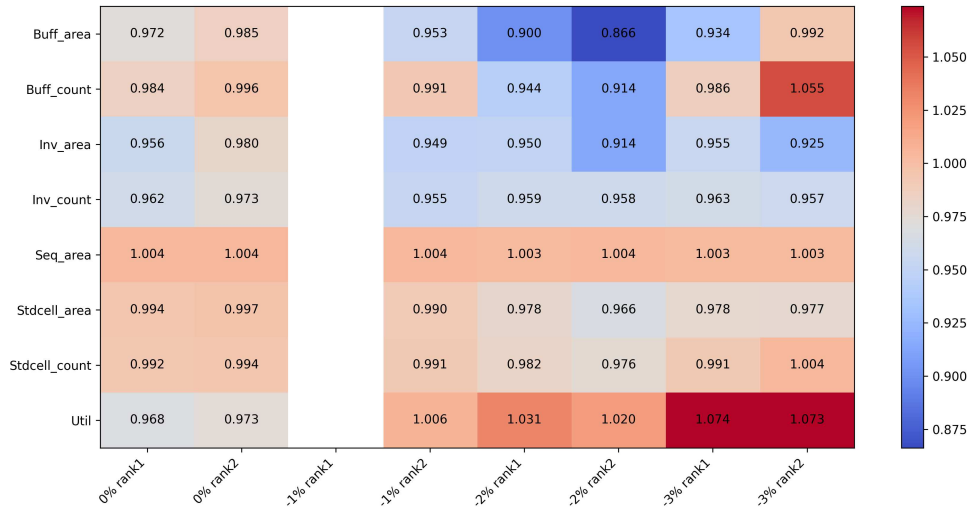



Figure 3.18: HeatMap QoR Post Route Cell Metrics.

In general, and consistent with the placement-stage results, all cell metrics remain close to the normalized value of the default configuration, indicating that the routing stage does not significantly alter the overall cell distribution or density.

### Wire Metrics Analysis

Figure 3.19 presents the wire-related metrics collected after the routing stage for each placement configuration. The metrics include Net count, Wirelength, DRC violations, and Shorts, providing a comprehensive view of the routing quality and physical feasibility of each layout.

As observed in the placement stage, the Net count remains consistently close to 1 across all configurations, confirming that the logical connectivity of the design is preserved throughout the flow. Similarly, the Wirelength metric stays very close to the normalized baseline, with a slight downward trend as the area is reduced; this behavior mirrors the trend seen post-placement and is expected, as more compact layouts naturally reduce the average interconnect distance. Notably, post-route wirelength values are slightly lower than those observed after placement, likely due to more accurate modeling of routing paths and detours during detailed routing.

An addition in the post-route analysis is the inclusion of DRC violations and Shorts, which provide direct insight into the physical correctness of the routed design. The most striking observation is the sharp increase in DRC violations for the most compact configurations; while moderate area reductions maintain DRC counts within acceptable limits, the -3% area reduction, particularly in rank 2, results in a dramatic spike in violations, over 100x higher than the default case. This trend is further confirmed by the Shorts metric, which also shows a substantial increase in the same configurations.

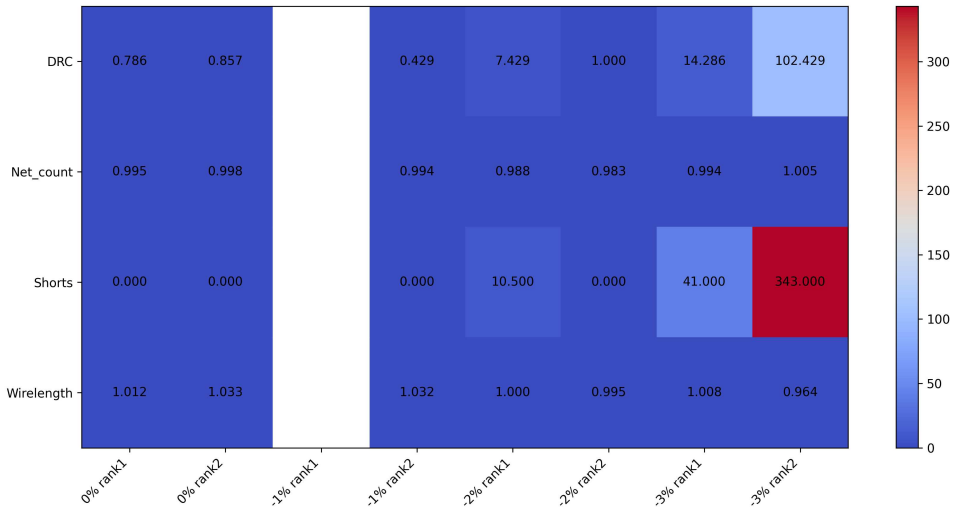



Figure 3.19: HeatMap QoR Post Route Wire Metrics.

These results highlight the limitations of aggressive area compaction: while wirelength and connectivity remain stable, the physical feasibility of the routing degrades significantly, leading to a high number of rule violations and shorts. This underscores the importance of balancing area optimization with routability and manufacturability constraints in advanced physical design flows.

### Power Metrics Analysis

Figure 3.20 presents the post-route power metrics across the various placement configurations. The metrics include internal power, leakage power, switching power, and total power, all normalized with respect to the default configuration.

As observed in the placement stage, the *internal power* remains remarkably stable, with values consistently close to 1 across all runs; this confirms that the internal power component, primarily driven by cell activity and capacitance, is largely unaffected by the routing stage.

The *leakage power*, on the other hand, shows a uniform increase across all configurations, ranging from approximately 2% to 6% above the default; this behavior is expected, as routing compaction and increased cell proximity can lead to higher leakage due to layout-dependent effects such as threshold voltage shifts and increased gate density.

The *switching power*, which was previously identical to the default across all configurations in the post-placement heatmap (Fig. 3.14), now exhibits slight variations. While some runs, such as 0% rank 1 and -3% rank 2, maintain the same switching power as before, others show a modest increase. The most notable case is 0% rank 2, which registers a 4.7% increase in switching power, possibly due to changes in net routing or increased coupling activity introduced during detailed routing.

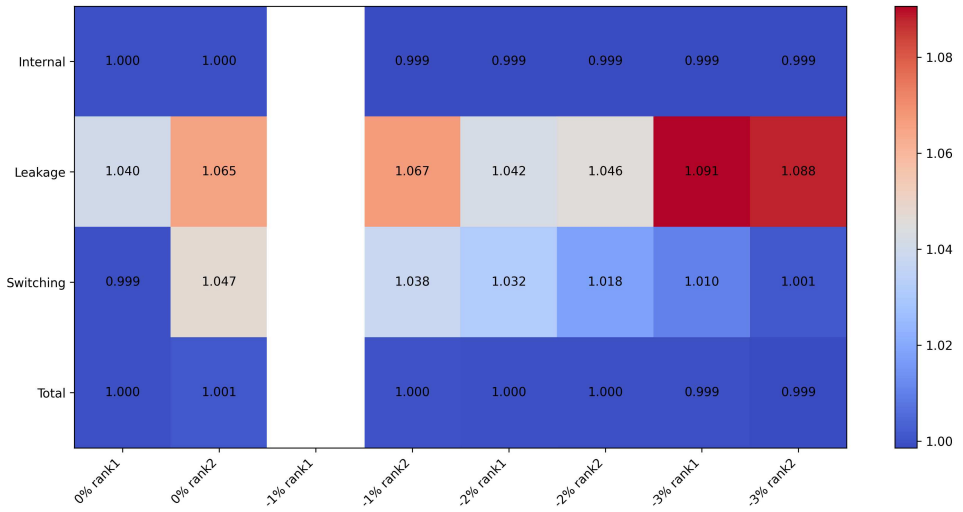



Figure 3.20: HeatMap QoR Post Route Power Metrics.

Despite these localized variations, the *total power* remains effectively unchanged across all configurations, mirroring the trend observed after placement and this indicates that the combined effects of internal, leakage, and switching power remain balanced, and that the routing stage does not introduce significant deviations in overall power consumption.

### Clock Metrics Analysis

Clock-related metrics weren't available after the placement stage since the clock tree synthesis (CTS) is performed subsequently. Figures 3.21 and 3.22 present two complementary heatmaps that provide insight into the quality and efficiency of the clock network across the various placement configurations.

In the first heatmap (Figure 3.21), we observe that the *total clock area* remains very close to the default value across all configurations; a similar trend is seen in the *clock wirelength*, which also remains close to 1. The only notable deviation is observed in the 0% rank 2 configuration, which shows a 7.5% increase in wirelength, suggesting a less efficient clock routing in that specific case.

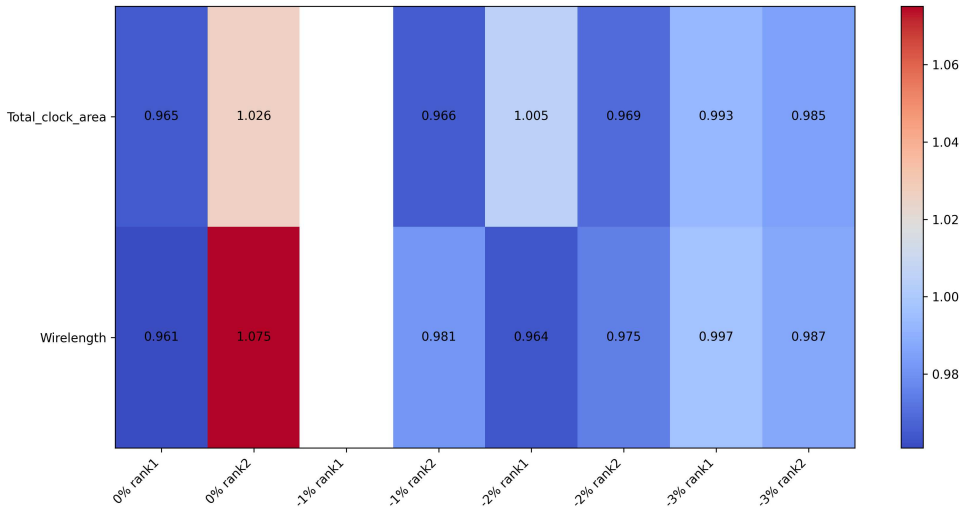



Figure 3.21: HeatMap QoR Post Route Clock Metrics.

The second heatmap (Figure 3.22) provides a more detailed breakdown of the clock tree structure and performance. Several observations can be made:

- *Buffer area* and *buffer count* remain very close to 1 in most configurations, indicating stable buffering requirements. The only exception is -3% rank 1, which shows a significant increase of nearly 50%, likely due to the need for additional buffering to compensate for increased congestion or skew.

- *Inverter area* and *inverter count* are also generally close to the default value, with the exception of 0% rank 1, which shows a 15% reduction. This is particularly interesting given that this same configuration performs worse in terms of total clock area and wirelength, suggesting a possible trade-off between inverter usage and routing complexity.

- *Clock skew* values are consistently higher than the default across all configurations, ranging from 12% up to 40% in the case of 11% rank 2. This indicates that tighter area constraints may introduce more variability in clock arrival times, requiring careful balancing.

- The *number of logic levels* remains very close to 1, though there is a slight upward trend as the area is reduced, suggesting a marginal increase in clock tree depth.

- The *maximum insertion delay*, which represents the longest delay from the clock source to any endpoint in the design, also remains close to the default across all configurations. This metric is critical for ensuring that the clock signal reaches all parts of the design within acceptable timing bounds.

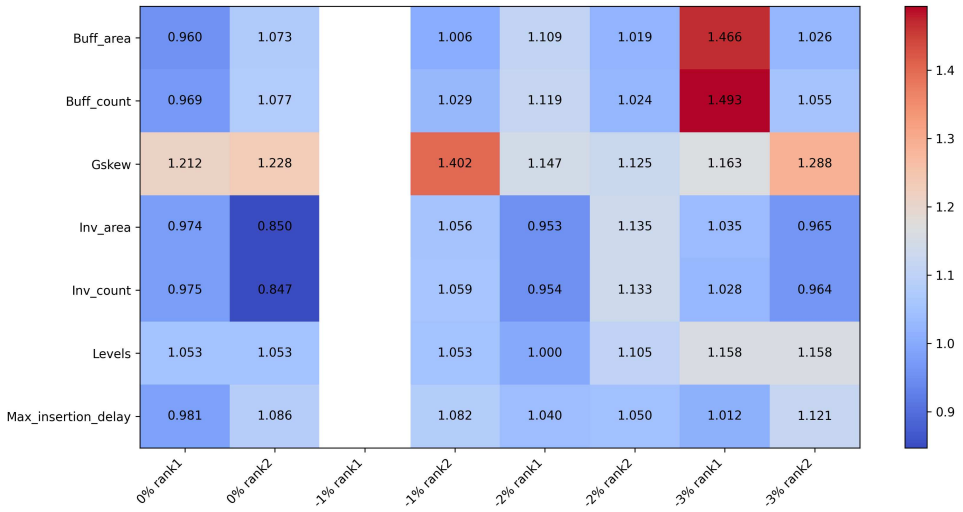



Figure 3.22: HeatMap QoR Post Route CLKM1_PROC Metrics.

In summary, the clock metrics confirm that the CTS process is generally robust across different placement strategies. While some configurations, particularly those with aggressive area reductions, require more buffering and exhibit higher skew, the overall structure and timing of the clock tree remain well-controlled.

### Timing Margin Analysis Across Corners

After completing the routing stage, a more comprehensive timing analysis is performed to evaluate the robustness of each configuration under a wider range of operating conditions. Compared to the placement-stage analysis, this post-route evaluation includes a larger set of PVT corners, capturing more realistic and extreme scenarios. Moreover, in addition to the setup timing analysis, we now also consider hold timing, providing a complete picture of the design's temporal behavior.

Fig. 3.23 presents the WNS for setup timing across seven corners. Several trends can be observed:

- *Corners 1 and 2* are clearly the least critical, as all configurations exhibit positive slack, however, there is notable variability among the different runs, suggesting that while timing closure is achieved, the margin can vary significantly depending on the placement strategy.

- *Corners 3 to 6* show small slack values, both positive and negative, but the differences between configurations are minimal. This indicates that these corners are moderately critical, but the placement strategies do not drastically affect timing in these conditions.

- *Corners 7 and 8* represent the most pessimistic scenarios, and all configurations exhibit negative slack; interestingly, the default configuration shows the highest WNS, while it tends to decrease as the area is reduced. However, this trend reverses slightly at the -3% area configurations, where WNS increases again, though it remains lower than the default.

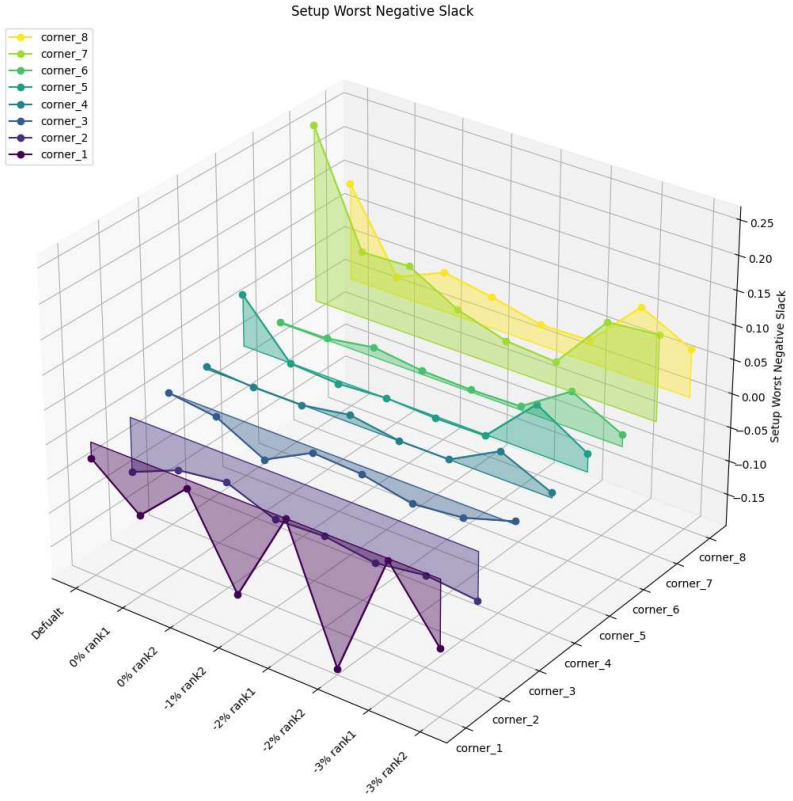



Figure 3.23: Timing QoR PostRoute Setup Worst Negative Slack.

Overall, the results suggest that there exists a sweet spot in area reduction where timing performance is optimized. Identifying and targeting this region is crucial for balancing area efficiency with timing robustness, especially in advanced design flows where aggressive compaction must be weighed against the risk of timing violations under worst-case conditions.

Complementing the WNS analysis, Fig. 3.24 illustrates the TNS across the same set of eight corners and placement configurations; as expected, the first six corners, characterized by either positive slack or values very close to zero, exhibit negligible TNS across all runs. This confirms that timing violations are either absent or extremely limited in these operating conditions, regardless of the placement strategy.

However, the behavior changes significantly in *Corners 7 and 8*, which represent

the most pessimistic scenarios in terms of process, voltage, and temperature. Here, the TNS increases markedly, revealing a more substantial accumulation of violating paths; the trend observed in the WNS analysis is largely preserved: the default configuration consistently performs worse than the optimized runs with 0% area reduction (both rank1 and rank2), as well as those with -1% and -2% area reduction.

Interestingly, a key divergence emerges when examining the -3% area configurations. Unlike the WNS case, where the -3% runs showed a slight degradation compared to -2%, yet still outperformed the default, the TNS values for -3% area are higher than those of the default configuration in both rank1 and rank2. This suggests that while the most aggressive area reduction may still improve the worst-case path, it introduces a larger number of violating paths overall, potentially due to increased congestion or suboptimal path distribution.

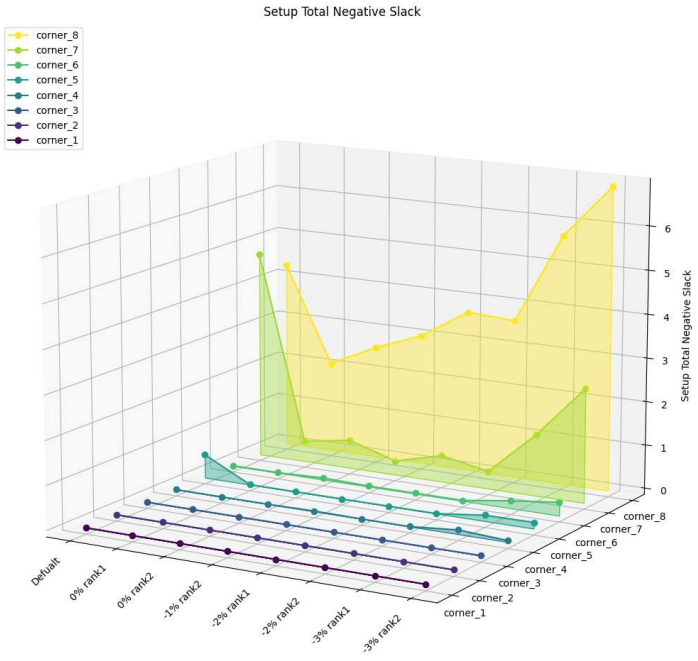



Figure 3.24: Timing QoR PostRoute Setup Total Negative Slack.

These findings underscore the importance of evaluating both WNS and TNS when assessing timing robustness. While WNS highlights the single most critical path, TNS provides insight into the cumulative impact of timing violations, which can be more indicative of the overall design health. In this context, moderate area reductions (up to -2%) appear to strike a better balance between compaction and timing integrity.

Finally, Fig. 3.25 reports the number of failure points observed across the seven corners and placement configurations. As anticipated from previous results, the first

six corners do not exhibit any significant timing violations, resulting in a failure point count that is effectively zero across all runs.

The situation changes notably in *Corners 7 and 8*, which correspond to the most pessimistic operating conditions; these corners show a substantial increase in the number of failure points, in line with the elevated TNS values previously discussed. In particular, *Corner 8* emerges as the most critical, with the number of failure points ranging from approximately 500 in the 0% area reduction rank1 configuration to over 1000 in the -3% area reduction rank1 case.

Interestingly, while the default configuration exhibits a higher TNS than the -2% area configurations, it actually results in a lower number of failure points; this suggests that although the default placement may have more severe violations in terms of cumulative slack, the violations are concentrated in fewer paths. In contrast, the -2% area configurations, despite having a lower TNS, show a broader distribution of violations across the design. On the other hand, the configurations with 0% area reduction and the single rank with -1% area consistently achieve both lower TNS and fewer failure points, confirming their superior timing robustness.

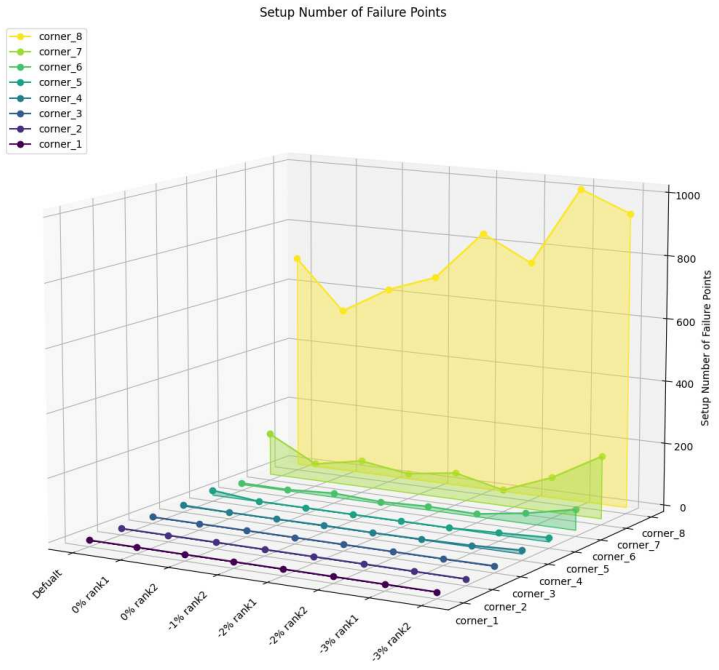



Figure 3.25: Timing QoR PostRoute Setup Number Failure Point.

This behavior highlights the importance of jointly analyzing TNS and failure point count: while TNS captures the total slack deficit, the number of failure points reveals how widespread the violations are. A configuration with fewer but deeper violations may be easier to fix than one with many shallow violations, which could require more extensive design changes.

While setup timing violations are typically the primary focus during timing closure, since they directly impact the maximum achievable clock frequency, **hold** timing must also be carefully evaluated to ensure functional correctness across all operating conditions. Unlike setup violations, which occur when data arrives too late, hold violations arise when data arrives too early, potentially leading to incorrect latching of values.

These violations are particularly critical under fast process, high voltage, and low temperature conditions, where signal propagation is accelerated and clock skew may be reduced. As a result, even short paths that are otherwise harmless under typical conditions can become problematic. Moreover, hold timing issues are often more difficult to fix post-route, as they may require the insertion of delay buffers or changes to the clock tree, which can disrupt previously optimized portions of the design.

Hold timing is typically evaluated only after the routing stage because hold analysis is highly sensitive to actual wire delays and parasitics, which are not accurately modeled during placement. Unlike setup timing, which can be reasonably estimated using idealized wire models, hold timing requires precise information about interconnect geometry and load capacitance, details that become available only after detailed routing.

Figure 3.26 illustrates the WNS for hold timing across eight PVT corners, ordered from the least to the most critical based on their speed characteristics. Corners with slow process (SS or TT), high temperature, and low voltage are less prone to hold violations, while corners with fast process (FF), low temperature, and high voltage represent the most challenging scenarios due to accelerated signal propagation.

Analyzing the general trend across all corners, it is evident that the default configuration consistently exhibits higher WNS values. This suggests that the default placement strategy is less effective in mitigating early data arrival issues.

The two configurations with -3% area reduction (rank1 and rank2) also show elevated WNS values, generally higher than those of the other runs. However, in many cases, their performance remains comparable to the default, indicating that the degradation introduced by aggressive area compaction is not excessively severe. Notably, -3% rank2 performs better than both -2% rank2 and -3% rank1, indicating that even within aggressive compaction strategies, specific placement variants can yield more favorable timing outcomes.

The -2% area rank2, which in some corners achieves the highest WNS among all runs, highlighting a potential weakness in this specific placement strategy. In contrast, rank1 -2% area remains more competitive, with WNS values closer to those of the -1% and 0% area reduction configurations.

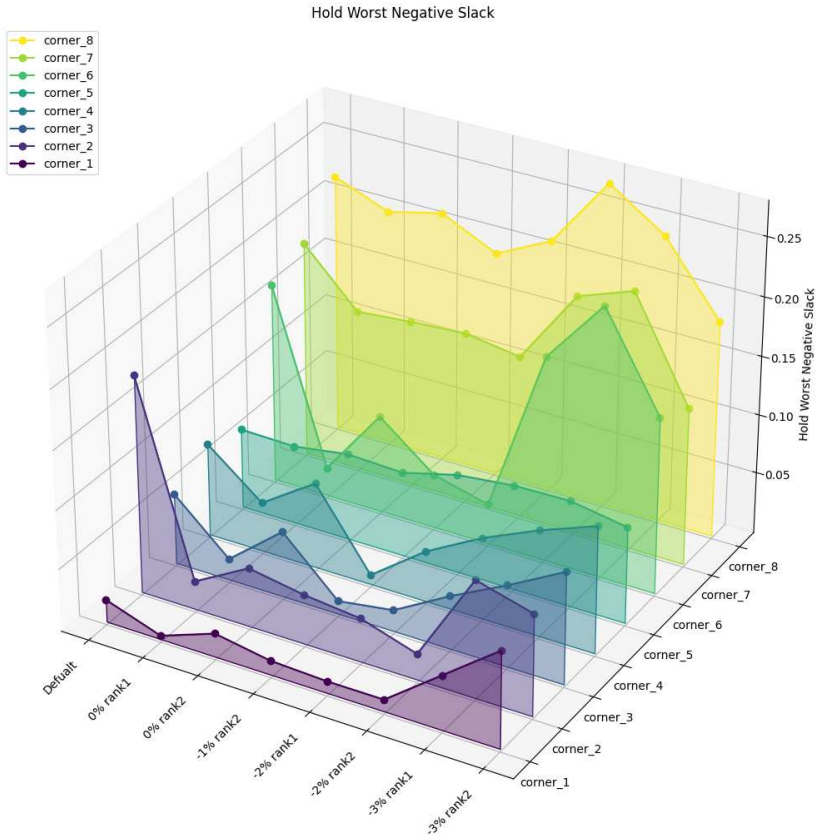



Figure 3.26: Timing QoR PostRoute Hold Worst Negative Slack.

The most robust results are observed in the 0% area reduction runs and the rank1 configuration at -1% area. These configurations consistently outperform the default, achieving lower WNS values across most corners; This confirms that even with slight area reductions, such as 0% or -1%, it is still possible to achieve excellent hold timing results, often outperforming the default configuration.

Figure 3.27 shows the TNS for hold timing. A striking observation is that the last corner, corresponding to the fastest and most critical operating conditions, exhibits a TNS significantly higher than all others; this suggests a widespread presence of hold violations in that scenario and anticipates a correspondingly high number of failure points.

Beyond this extreme case, the general trend reveals several interesting patterns. Despite often showing higher WNS values, the default configuration maintains TNS values that are largely comparable to those of the 0% and -1% area reduction runs. This mirrors the behavior observed in the setup timing analysis, where the default also concentrated violations in fewer but deeper paths.

The -3% rank1 configuration clearly performs the worst, with consistently high TNS values across multiple corners; this aligns with its poor WNS performance and

confirms that aggressive area reduction in this case leads to a widespread degradation in hold timing.

Surprisingly, the -3% rank2 configuration performs better than expected; while it still shows degradation in some corners, in the most critical ones it achieves TNS values comparable to the default configuration. This partially confirms the trend already hinted at in the WNS analysis and suggests that, under certain conditions, even aggressive compaction can be mitigated by a more favorable placement strategy.

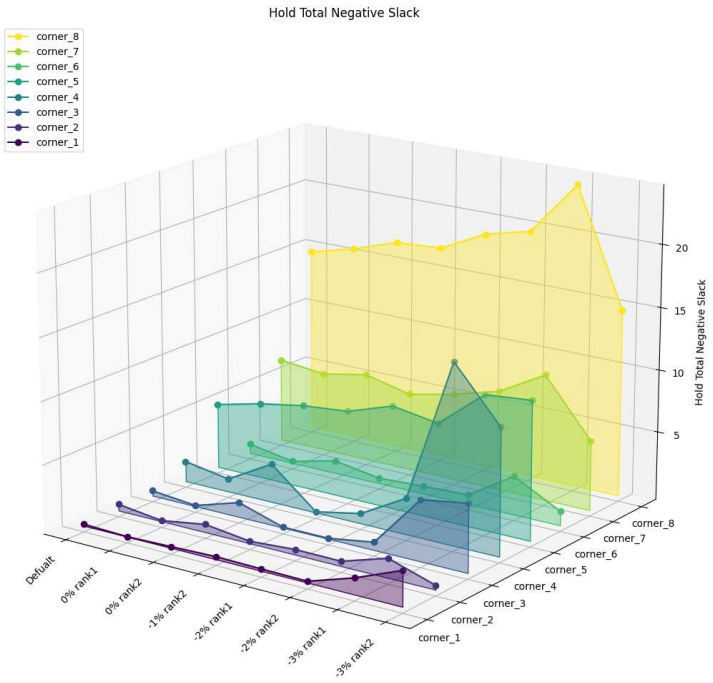



Figure 3.27: Timing QoR PostRoute Hold Total Negative Slack.

The analysis confirms that moderate area reductions (0% and -1%) offer a good trade-off, while more aggressive reductions require careful tuning to avoid widespread hold failures.

Figure 3.28 presents the number of failure points for hold timing across and, as expected, the overall trends closely mirror those observed in the TNS analysis, confirming the correlation between the extent of slack violations and the number of affected paths.

However, a notable exception emerges in *corner 6*, which, despite not being the worst in terms of TNS, shows by far the highest number of failure points across all configurations. This indicates that, in this specific corner, violations are more widespread but individually less severe, resulting in a high number of failure points but relatively moderate TNS. Such behavior highlights the importance of analyzing both metrics in tandem, as they capture complementary aspects of timing robustness.

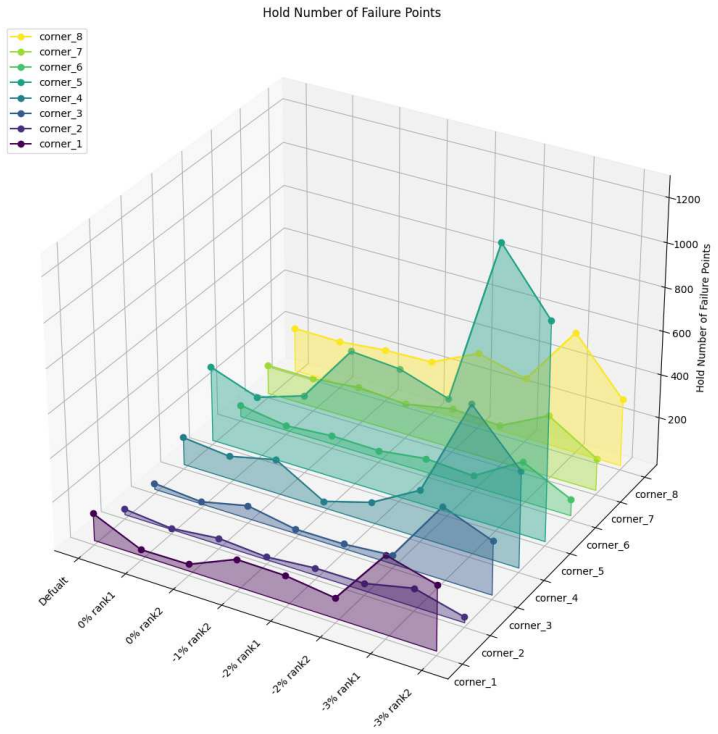



Figure 3.28: Timing QoR PostRoute Hold Number Failure Point.

The post-route stage represents a critical checkpoint in the physical design flow, where the impact of detailed routing, parasitics, and congestion becomes fully visible. Through a comprehensive analysis of cell, wire, power, clock, and timing metrics, this section has highlighted the strengths and limitations of various placement strategies under realistic operating conditions.

From a physical standpoint, moderate area reductions (up to -2%) maintain stable cell distributions and wirelengths, while avoiding the severe DRC violations and shorts observed in the most compact configurations; **power metrics** remain largely consistent across all runs, with only minor variations in leakage and switching components, finally, **clock metrics** confirm the robustness of the CTS process, though tighter area constraints tend to increase skew and buffering requirements.

The timing margin analysis across multiple PVT corners has revealed distinct trade-offs between setup and hold robustness. For **setup timing**, configurations with 0% to -2% area reduction consistently outperform the default, achieving better slack and fewer violations. For **hold timing**, the behavior is more complex: while the default often shows higher WNS, its TNS and failure point counts remain competitive, suggesting concentrated but less widespread violations. Notably, the -3% rank1 configuration performs poorly across all timing metrics, whereas -3% rank2 shows surprisingly resilient behavior in critical corners.

Overall, Moderate area reductions offer the best balance between physical feasibility and timing integrity while aggressive compaction may yield benefits in specific scenarios but introduces significant risks that must be carefully managed

## 3.3 Conclusion

The increasing complexity of modern integrated circuits, driven by the relentless pace of Moore's Law, has placed unprecedented demands on the **physical design** phase of the VLSI flow. Among the most critical and challenging tasks in this domain is **macro placement**, a process that remains largely manual and heuristic-driven despite its significant impact on downstream stages such as routing, timing closure, and manufacturability; as highlighted in Chapter 1, this lack of automation limits the ability to explore alternative floorplan configurations and obstruct the scalability of design methodologies in the face of growing SoC complexity.

To address this gap, this thesis has investigated the integration of **machine learning techniques** into the macro placement process, with the goal of enabling fast, flexible, and automated **floorplan exploration**. Chapter 2 provided a theoretical and algorithmic foundation for this effort, reviewing the evolution of placement strategies from stochastic and partitioning-based methods to modern analytical placers; particular attention was given to **mixed-size placement**, where the coexistence of standard cells and large macros introduces unique optimization challenges. Within this context, the thesis explored the capabilities of state-of-the-art engines such as ePlace [1], ePlace-MS [3], and DREAMPlace [2], all of which leverage electrostatics-inspired density models and, in the case of DREAMPlace, GPU-accelerated deep learning frameworks.

Building on these foundations, the experimental contribution presented in Chapter 3 focused on extending a proprietary macro placement tool developed at **Qualcomm®**; this tool, originally designed to generate diverse macro placements using a machine learning engine, was enhanced with a GUI and integrated with an **area shrink** optimization engine. The resulting framework enables designers to explore multiple floorplan configurations under varying area constraints, while maintaining control over key physical and timing metrics.

The test campaign demonstrated the tool's ability to generate and evaluate eight distinct macro placements across four area configurations (0%, -1%, -2%, -3%) in parallel; notably, the entire exploration process was conducted by a single engineer with limited prior experience, highlighting the accessibility and efficiency of the pro-

posed framework. The analysis covered both qualitative and quantitative aspects, including cell utilization, wirelength, congestion, power consumption, and timing margins across multiple PVT corners. The results revealed several key insights:

- Moderate area reductions (up to -2% of total area shrink that corresponds to a 7.26% reduction in placeable core area, as seen in Tab. 3.1) consistently yielded better or comparable QoR than the default configuration, both in terms of physical feasibility and timing robustness.

- Aggressive compaction (-3% that coincide with a placeable core area ahrink of 10.84%, as seen in Tab. 3.1) introduced significant risks, including increased DRC violations, higher congestion, and degraded timing, particularly in the most pessimistic corners.

These results not only validate the effectiveness of the proposed toolchain but also reinforce the broader thesis that machine learning-driven automation can significantly enhance early-stage design exploration, particularly in the context of macro placement. By enabling the generation and evaluation of multiple floorplan alternatives in parallel, the tool empowers designers to make more informed architectural decisions, reduce iteration cycles, and ultimately improve the overall quality of the final layout.

At the same time, the study has highlighted some limitations and areas for future improvement, for instance, the current version of the macro placer does not guarantee legal placements, requiring manual intervention in the legalization phase. Additionally, while the tool supports area shrink exploration, it does not yet incorporate timing-driven placement during macro generation. Addressing these aspects could further enhance the tool's applicability in industrial flows.

Looking ahead, several promising directions emerge:

- *Automatic legalization*: Integrating legality checks and fixed-cell awareness directly into the placement engine would reduce manual effort and improve flow automation.

- *Timing-driven macro placement*: Incorporating timing feedback during macro generation could help avoid configurations that are physically feasible but timing-critical.

- *Multi-objective optimization* beyond wirelength and congestion: Expanding the Pareto evaluation to include power, thermal, and manufacturability metrics would provide a more comprehensive view of design trade-offs.

- *Reinforcement learning*: Leveraging reinforcement learning could enable the tool to learn from past placements and improve its decision-making over time.

In conclusion, this thesis has focused on the development and evaluation of a framework for **macro placement exploration**, integrating a machine learning-based placer with area shrink capabilities and a graphical interface. The proposed solution was tested across multiple area configurations and evaluated using a comprehensive set of physical and timing metrics. The results confirm that moderate area reductions can improve or preserve design quality compared to the default configuration, while aggressive compaction introduces significant risks. The tool has demonstrated its effectiveness in supporting early-stage floorplan exploration, enabling the generation of diverse and viable macro placements with minimal manual effort.

# Bibliography


[1] Jingwei Lu, P. Chen, Chin-Chih Chang, Lu Sha, Dennis Huang, Chin-Chi Teng, and Chung-Kuan Cheng. eplace: Electrostatics-based placement using fast fourier transform and nesterov's method. *ACM Transactions on Design Automation of Electronic Systems*, 20, 02 2015.

[2] Yibo Lin, Zixuan Jiang, Jiaqi Gu, Wuxi Li, Shounak Dhar, Haoxing Ren, Brucek Khailany, and David Z. Pan. Dreamplace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(4):748–761, 2021.

[3] Jingwei Lu, Hao Zhuang, Pengwen Chen, Hongliang Chang, Chin-Chih Chang, Yiu-Chung Wong, Lu Sha, Dennis Huang, Yufeng Luo, Chin-Chi Teng, and Chung-Kuan Cheng. eplace-ms: Electrostatics-based placement for mixed-size circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(5):685–698, 2015.

[4] G.E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.

[5] Guyue Huang, Jingbo Hu, Yifan He, Jialong Liu, Ma Mingyuan, Zhaoyang Shen, Juejian Wu, Yuanfan Xu, Hengrui Zhang, Xuefei Ning, Yuzhe Ma, H.Y. Yang, Bei Yu, Huazhong Yang, and Yu Wang. Machine learning for electronic design automation: A survey. 01 2021.

[6] Andrew B. Kahng. Machine learning applications in physical design: Recent results and directions. In *Proceedings of the 2018 International Symposium on Physical Design*, ISPD '18, page 68–73, New York, NY, USA, 2018. Association for Computing Machinery.

[7] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nova, Jiwoo Pak, Andy Tong, Kavya Srinivasa, William Hang, Emre Tuncer, Quoc V. Le, James Laudon, Richard Ho, Roger Carpenter, and Jeff Dean. A

graph placement methodology for fast chipdesign. *Nature*, 594(7862):207–212, Jun 2021.

[8] Changyong Oh, Roberto Bondesan, Dana Kianfar, Rehan Ahmed, Rishubh Khurana, Payal Agarwal, Romain Lepert, Mysore Sriram, and Max Welling. Bayesian optimization for macro placement, 2022.

[9] Sivaramakrishnan Harihara Subramanian, Khris M Valencia Chacon, and Venkatesh R S. An industrial perspective on ml-macro placement methods: Challenges and recommendations. In *Proceedings of the Great Lakes Symposium on VLSI 2025*, GLSVLSI '25, page 527–533, New York, NY, USA, 2025. Association for Computing Machinery.

[10] Andrew B. Kahng, Jens Lienig, Igor L. Markov, and Jin Hu. *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Springer Dordrecht, 2011.

[11] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: applications in vlsi domain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 7(1):69–79, 1999.

[12] C.M. Fiduccia and R.M. Mattheyses. A linear-time heuristic for improving network partitions. In *19th Design Automation Conference*, pages 175–181, 1982.

[13] Rupesh S. Shelar and Marek Patyra. Impact of local interconnects on timing and power in a high performance microprocessor. In *Proceedings of the 19th International Symposium on Physical Design*, ISPD '10, page 145–152, New York, NY, USA, 2010. Association for Computing Machinery.

[14] C. Sechen and A. Sangiovanni-Vincentelli. Timberwolf3.2: A new standard cell placement and global routing package. In *23rd ACM/IEEE Design Automation Conference*, pages 432–439, 1986.

[15] J.A. Roy, S.N. Adya, D.A. Papa, and I.L. Markov. Min-cut floorplacement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(7):1313–1326, 2006.

[16] Natarajan Viswanathan, Min Pan, and Chris Chu. *FastPlace: An Efficient Multilevel Force-Directed Placement Algorithm*, pages 193–228. Springer US, Boston, MA, 2007.

[17] Myung-Chul Kim, Dong-Jin Lee, and Igor L. Markov. Simpl: An effective placement algorithm. In *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 649–656, 2010.

[18] Chung-Kuan Cheng, Andrew B. Kahng, Ilgweon Kang, and Lutong Wang. Replace: Advancing solution quality and routability validation in global placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(9):1717–1730, 2019.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.