

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

CLUSTERING TECHNIQUES FOR FLOORPLANNING LARGE-SCALE MIXED-SIZE CIRCUITS

ALFONSO BROWN GONZÁLEZ

Thesis supervisor
JORDI CORTADELLA FORTUNY (Department of Computer Science)

Degree
Master's Degree in Innovation and Research in Informatics (Advanced Computing)

Master's thesis

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Abstract

The continuous scaling of semiconductor technology has introduced extreme structural diversity in VLSI design, making the placement of millions of standard cells and large-scale macros at the same time increasingly more complicated. Traditional methods that completely decouple macro floorplanning from standard cell placement often result in sub-optimal wirelength and routing congestion, and often involve a lot of manual iterations. This thesis proposes a hierarchical, connectivity-driven placement framework for large-scale mixed-size circuits. By modeling the electrical netlist as a hypergraph and applying weight normalization strategies to isolate topological affinity, the proposed framework partitions the circuit into hybrid blocks with both macros and standard cells mixed in while testing with varying parameters for imbalance ϵ and number of partitions k. Then the methodology leverages state-of-the-art floorplanning algorithms to fix the coordinates, navigate constraints, and legalize the design of the blocks. Finally it tests a center-based strategy and a pull-force-driven (PACCS) strategy for displacing macros, prior to using a state-of-the-art placement algorithm for standard cells, via Nesterov's Accelerated Gradient. Experimental validation using ISPD 2005 and 2006 benchmarks demonstrates that intermediate hypergraph partitioning configurations ($k > 0$) and higher imbalances ($\epsilon \geq 0.6$) optimize the balance between logical cohesion and physical flexibility, producing highly competitive Half-Perimeter Wirelength (HPWL) metrics. Furthermore, while advanced routing-aware metrics like KM-1 and SOED correlate more closely with layout quality than standard Edge-cut, we observe that none serve as absolute predictors. Ultimately, this work proves that guiding analytical optimization engines with topology-driven pre-clustering effectively manages the complexity of modern mixed-size placement.

Acknowledgments

I would like to express my deepest gratitude to my thesis supervisor, Jordi Cortadella Fortuny, for suggesting this compelling research topic and for his guidance, patience, expertise, and for providing me access into his floorplanning tool and research.

I am also incredibly thankful for the companionship and support of my friends from the MIRI master's program; with special thanks to Virginia for always being there. Finally, I want to express my profound thanks to my family. To my parents, Elsa and Alfonso, and my siblings, Lorena and Leonardo, thank you for your endless support throughout my professional and academic life; and to my cousins, Natalia and Amalia, thank you for your companionship during my time in Barcelona.

Contents

1 Introduction **1**
- 1.1 Background and Motivation . . . 1
- 1.2 Problem Statement . . . 3
- 1.3 Proposed Work and Objectives . . . 4
  - 1.3.1 Hypotheses . . . 4

2 Theoretical Framework **6**
- 2.1 The VLSI Physical Design Flow . . . 6
- 2.2 Floorplanning . . . 7
  - 2.2.1 Slicing vs. Non-Slicing Floorplans . . . 8
  - 2.2.2 Polish Notation . . . 8
  - 2.2.3 Sequence Pairs . . . 9
- 2.3 Netlist Representation as Hypergraphs . . . 10
  - 2.3.1 Hypergraph Partitioning . . . 10
- 2.4 Wirelength Estimation Models . . . 12
  - 2.4.1 Differentiable Approximations . . . 12
- 2.5 Analytical Global Placement . . . 12
  - 2.5.1 The Electrostatic Density Analogy . . . 12
  - 2.5.2 Nesterov's Accelerated Gradient . . . 13

3 State of the Art **14**
- 3.1 Hypergraph Partitioning and Clustering . . . 14
- 3.2 Floorplanning & Mixed-Size Placement . . . 15
- 3.3 Industry Standards . . . 16

4 Methodology **18**
- 4.1 Benchmarks pre-processing . . . 18
- 4.2 Netlist Hypergraph Modeling . . . 20
- 4.3 Partitioning and Metric Extraction . . . 22
- 4.4 Block Reconstruction and Floorplanning . . . 23
  - 4.4.1 Block Reconstruction . . . 23
  - 4.4.2 Floorplanning . . . 24
- 4.5 Macros Placement . . . 25
- 4.6 Global Placement . . . 27

4.7 HPWL Measurement 28

5 Results **29**
5.1 Hypergraph Partitioning 29
5.2 Floorplanning 31
5.3 Placement 33

6 Discussion **38**
6.1 Evaluation of the Placement Framework 38
6.2 Hypergraph Partitioning and Metric Tuning 38
6.3 Mixed-Sized Placement 39

7 Conclusions **41**

8 Future Work **43**
8.1 Hypergraph Partitioning 43
8.2 Floorplanning Configurations 44
8.3 PACCS Refinement 45
8.4 Standard Cells Incremental Placement 46

Appendices **52**

A Benchmarks' Original Placements **53**

B Full measured HPWL **56**

C Best Found Placement for every Benchmark **60**

List of Figures

1.1 Mixed-Size Placement Problem 1
1.2 VLSI Design Flow 2
1.3 Mixed-Size Problem 3

2.1 Slicing & Non-Slicing Floorplans 7
2.2 Polish Expression Example 8
2.3 Horizontal and Vertical Constraint Graphs 9
2.4 Sample hypergraph. 11

3.1 Circuit - Hypergraph bipartition. 15

4.1 Methodology flow. 18
4.2 I/O pins displacement 20
4.3 Net Size Disparity 20
4.4 Standard Cells Distribution 22
4.5 Partitioned hypergraph 23
4.6 Sample Floorplanning Process 24
4.7 Macros placement within blocks. 27

5.1 Hypergraph metrics behavior 29
5.2 KM-1 degradation 30
5.3 CV Depends on Benchmark Design. 31
5.4 Failed floorplanning for *newblue3* 32
5.5 Normalized HPWL vs k & vs ϵ 33
5.6 Partition Metrics Correlation with HPWL for each Benchmark. . 34
5.7 Center placement vs PACCS 35
5.8 Issue when placing Standard Cells. 36

8.1 Multiple rectangle floorplanning 44
8.2 Multiple rectangle die 45

A.1 Original Placement for ISPD 2005 Benchmarks. 54
A.2 Original Placement for ISPD 2006 Benchmarks. 55

C.1 Best Found Placement for ISPD 2005 Benchmarks. 61
C.2 Best Found Placement for ISPD 2006 Benchmarks. 62

List of Tables

4.1 Benchmarks Description 21
4.2 Benchmarks - Macros Classification 21

5.1 Connectivity Metrics Correlation 30
5.2 Failed Floorplans 32
5.3 Average Normalized Partition Metrics Correlation 34
5.4 Best Found HPWL for each Benchmark. 36
5.5 HPWL Comparison with other Placers. 37

B.1 Final HPWL - Macros at Center 58
B.2 Final HPWL - Using PACCS 59

Chapter 1

Introduction

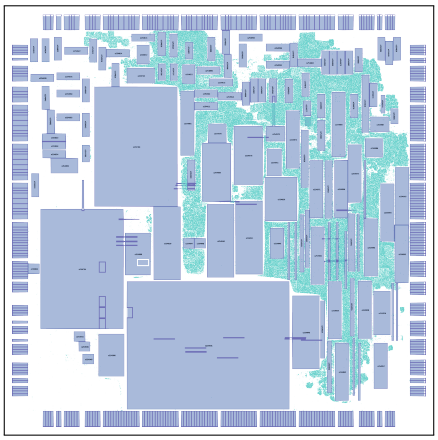


Figure 1.1: Final placement of macros and standard cells in a modern mixed-size placement problem.

1.1 Background and Motivation

The rapid advancement of semiconductor technology has transitioned the industry into the deep sub-micron and nanometer eras. This scaling has enabled the integration of billions of transistors onto a single silicon die, facilitating the rise of highly complex System-on-Chip (SoC) architectures. Modern electronics are no longer characterized by simple homogeneous arrays of uniform transistors and logic gates, instead as seen in Figure 1.1 they have a very varied landscape with components of all sizes.

In a typical contemporary circuit design, there are millions of small logic gates, called standard cells, that must coexist with large-scale components known as macros. These macros include Intellectual Property (IP) cores, embed-

ded memories (SRAMs), digital signal processors (DSPs), and various analog interfaces. The physical dimensions of these macros can be several orders of magnitude larger than a single standard cell, creating a big disparity in the layout granularity.

This increased structural diversity and scale means that designing modern integrated circuits has long bypassed traditional engineering methodologies, and relies less on human intervention. This gave rise to entire dedicated fields of study: Very Large Scale Integration (VLSI) design and Electronics Design Automation (EDA). To make the creation of these complex systems manageable VLSI design flow is broken down into multiple distinct, highly specialized phases, going from very high-level architectural specification, passing through logical minimization to reduce the amount of logic gates and transistors, and down to exact physical implementation. Each stage, as shown in Figure 1.2, aims to solve a specific subset of geometric and electrical challenges. Then, EDA encompasses all the tools used within the flow, including the software algorithms and mathematical solvers required to translate high-level logical descriptions into manufacturable physical geometries.

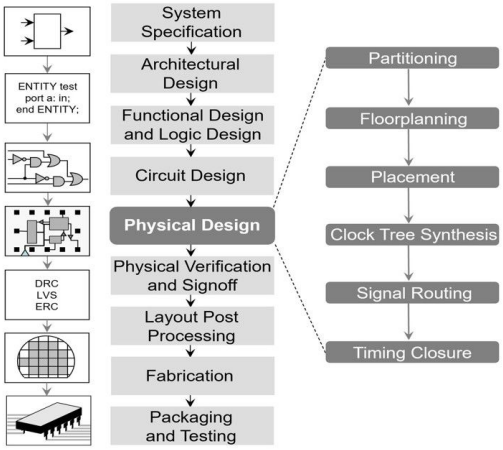


Figure 1.2: Diagram showing the common stages of VLSI Design (Taken from [1]).

Within these stages, the placement phase of the physical design flow is the process of determining the exact spatial coordinates for every component on the die. Placement is often considered one of the most critical bottlenecks in achieving design closure. The spatial arrangement of all components directly dictates the total wirelength, power consumption, signal integrity, and the ultimate timing performance of the integrated circuit. Still, because macros are much larger and pin-dominant, they also tend to have a greater impact on the circuits' performance. As complexity of macros keep growing and as wire delay has surpassed gate delay as the dominant factor in performance [2], the efficiency

of placement has become the primary determinant of chip quality.

Today, the industry often still relies on experienced engineers to correctly place the macros, requiring iterations of several weeks. Accurate timing information is usually only available after more detailed placement and routing stages. This results in very long and ineffective processes that involve a lot of repetitive rework. If some information of these later stages was available early and could help influence some of the design decisions, the whole process could become more efficient.

1.2 Problem Statement

Mixed-size circuit placement (as depicted in Figure 1.3) is the process of defining the spatial coordinates of standard cells and macros, roughly at the same time. It presents a unique and computationally intensive challenge that placement methodologies struggle to address effectively. The difficulty arises from the fundamental requirement to solve distinct optimization problems simultaneously. As thoroughly discussed in [3], some of these problems have "optimal" solutions for small instances, but have had trouble scaling to follow Moore's Law. Some others have a large solution space with great local minima, so require a lot of randomization to perform hill climbing. Ultimately, as further discussed in Chapter 3, research and solvers commonly focus on solving mixed-size placement as the composition of two main problems:

1. **Global Floorplanning:** The high-level arrangement of massive macro blocks to ensure they do not overlap while leaving sufficient "white space" for interconnects.
2. **Local Placement:** The fine-grained optimization of millions of standard cells to minimize wirelength and congestion.

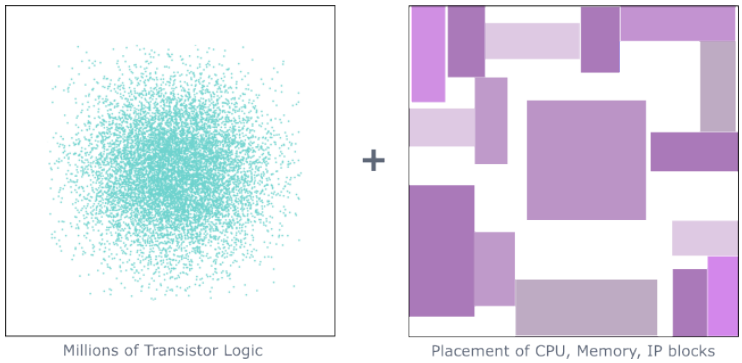


Figure 1.3: Depiction of the Mixed-Size Placement problem.

Traditional tools often treat these as decoupled steps, either manually fixing macros early in the flow or treating macros as oversized cells, which may block

optimal routes or lead to overlap and legalization issues. This disparity often results in sub-optimal wirelength, significant routing congestion in the channels between macros, and multiple costly iterations in the design process to meet timing constraints. While there is a lot of research in EDA tools that can improve the automation of these designs, the industry generally still prefers low-risk flows that involve manual intervention. This outlines the need for integrated, stable approaches that recognize the logical relationship between components, and that require less iteration.

1.3 Proposed Work and Objectives

This research proposes a hierarchical placement framework designed to include logical netlist connectivity information into physical floorplanning. Unlike other mixed-size placement methods that process macros and cells in fully isolated phases, or model it as a single placement problem, this approach leverages the connectivity of the circuit to cluster standard cells with their most strongly associated macros.

By modeling the circuit as a hypergraph and utilizing multi-way partitioning, we transform the mixed-size placement problem into a structured block-based floorplanning problem. The macros within these blocks are then fixed, and the global placement of the standard cells should take advantage of these new distributions. The primary objectives of this thesis are:

- **Connectivity-Driven Clustering:** To develop a strategy that clusters standard cells into blocks anchored by macros, ensuring that high-affinity logic remains physically proximal to the relevant blocks.
- **Hypergraph Analysis:** To evaluate the effectiveness of various partitioning metrics (Edge-cut, $k-1$, and SOED) in predicting the quality of the final physical layout.
- **Constraint-Aware Floorplanning:** To use existing floorplanning algorithms to handle these reconstructed hybrid blocks, under fixed-outline and aspect ratio constraints.
- **Region-Aware Standard Cell Placement:** To use existing placement algorithms to leverage the defined partitions and their regions to perform the placement of all the standard cells.
- **Benchmark Validation:** To validate the proposed methodology using industry-standard VLSI benchmarks, comparing the results against other mixed-size placement flows.

1.3.1 Hypotheses

The main hypothesis of this work is:

- **$\mathbf{H}_1$:** A hierarchical placement framework that transforms a normalized netlist hypergraph into hybrid macro-standard cell blocks and takes advantage of connectivity information can achieve comparable or lower HPWL compared to other placement flows.

Regarding hypergraph partitioning and its parameter tuning, and the measured metrics, the hypotheses are:

- **$\mathbf{H}_{2a}$:** The relationship between the target partition parameter k and the final HPWL is non-monotonic; specifically, there is an intermediate configuration ($k > 0$) that isolates clusters of highly connected standard-cell that will outperform a strict macro-anchored-only configuration ($k = 0$).

- **$\mathbf{H}_{2b}$:** Increasing the allowable area imbalance (ϵ) during the multi-way partitioning phase relaxes topological constraints, allowing the floorplanner more flexibility to satisfy fixed-outline and aspect ratio constraints, which reduces the complexity of final overlap legalization penalties.

- **$\mathbf{H}_{2c}$:** Advanced hypergraph partitioning objectives that account for net pins, such as Sum of External Degrees (SOED) and KM-1, will be more accurate predictors of the quality in the final physical layout (HPWL) than the standard Edge-cut metric.

Regarding floorplanning and standard cells placement:

- **$\mathbf{H}_{3a}$:** Placing macros at the center of the pre-partitioned hybrid blocks will result in a worse quality placement compared to displacing the macros according to the pull forces of the standard cells and their pin locations.

- **$\mathbf{H}_{3b}$:** Guiding the analytical global placement of standard cells using the pre-defined regions derived from legalized hybrid blocks reduces the continuous optimization search space, resulting in fewer Nesterov gradient descent iterations, faster convergence, and a lower final HPWL compared to a flat, unconstrained global placement flow.

Chapter 2

Theoretical Framework

The physical design of modern VLSI circuits is a highly complex, multi-objective optimization problem. In able to navigate the methodology proposed in this work as well as the and the state-of-the-art discussed, it is essential to establish a shared theoretical foundation. This chapter outlines the core mathematical models, graph theory concepts, and continuous optimization paradigms that underpin modern mixed-size placement algorithms.

2.1 The VLSI Physical Design Flow

The transition from a logical Register-Transfer Level (RTL) description to a physical layout ready for manufacturing (GDSII) is executed through the physical design flow. Due to the extreme computational complexity of placing billions of transistors, the flow is generally divided into distinct stages (Figure 1.2). Within Physical Design process, the stages are:

1. **Partitioning:** An initial planning stage where affine components are grouped together to reduce the problem size and simplify floorplanning.
2. **Floorplanning:** The initial placing stage where the die size is determined, I/O terminals are assigned, and large, rigid intellectual property (IP) blocks (macros) are placed.
3. **Placement:** The assignment of physical locations to standard cells within the core area. This is typically split into *Global Placement* (which finds approximate locations while minimizing wirelength and density) and *Detailed Placement* (which legally aligns cells to standard rows and prevents microscopic overlaps).
4. **Clock Tree Synthesis (CTS):** The physical wiring of the clock signals to all the sequential elements, while minimizing clock skew and power.

5. **Routing:** The physical placement of wires to connect components according to the netlist. It is also typically split into *Global Routing* (which assigns nets into routing regions without exact wire geometries) and *Detailed Routing* (which determines exact metal tracks and vias within each region, respecting DRC rules).

6. **Timing Analysis & Optimization:** Static timing analysis (STA) identifies critical paths. Gate sizing, buffer insertion, and wire optimization are applied iteratively to close timing.

The integration of massive macros alongside millions of tiny standard cells creates the *Mixed-Size Placement Problem*, which looks into solving both the Floorplanning and Placement stages while leveraging information usually reserved for future design decisions.

2.2 Floorplanning

Floorplanning is the foundational step of the physical design flow that goes from an abstract logical netlist into a concrete spatial floorplan. Unlike standard cell placement, which deals with millions of uniform, microscopic components, floorplanning focuses primarily on the arrangement of large, rigid, or flexible macro blocks (such as IP cores, memories, and DSPs). The primary objectives of floorplanning include minimizing the total die area (or bounding box), minimizing estimated wirelength, optimizing the chip's aspect ratio, and determining the locations of I/O pads.

To explore the solution space of possible block arrangements efficiently via optimization heuristics (e.g., Simulated Annealing), floorplans are modeled using mathematical representations. These representations are fundamentally categorized based on the geometric structures they can express: slicing and non-slicing floorplans.

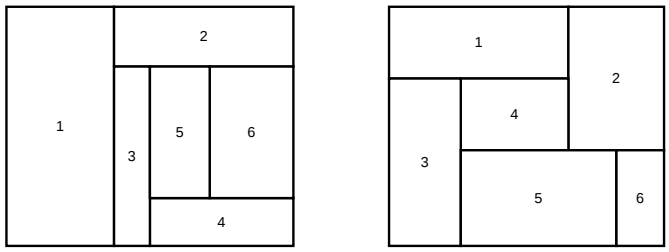


Figure 2.1: Comparison of a slicing floorplan to the left, and a non-slicing floorplan to the right.

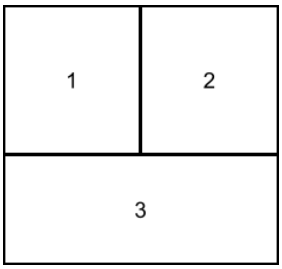


Figure 2.2: Example of a slicing floorplan with Polish Expression $1\,2\,V\,3\,H$.

2.2.1 Slicing vs. Non-Slicing Floorplans

The structural topology of a floorplan dictates the complexity of its representation and the algorithms required to optimize it:

- **Slicing Floorplan:** A floorplan is considered slicing if it can be obtained by recursively dividing a rectangle into smaller sub-rectangles using continuous horizontal or vertical cutting lines. Slicing floorplans are highly structured and can be naturally represented as a binary tree (a slicing tree), where internal nodes represent the cut directions and leaf nodes represent the modules.
- **Non-Slicing Floorplan:** A non-slicing floorplan contains structures that cannot be divided by a single continuous straight line sweeping from one edge to the other. Non-slicing configurations have greater flexibility and can achieve higher packing densities, but they require significantly more complex representation models and optimization algorithms.

2.2.2 Polish Notation

For slicing floorplans, the Polish Notation (or postfix notation) provides a compact, algebraic representation that is very practical for stochastic optimization methods like Simulated Annealing [4].

A Polish expression is derived from the post-order traversal of a slicing tree. For a design with n blocks, the expression consists of n operands (representing the blocks) and $n-1$ operators:

- V **(Vertical Cut):** Specifies that the left operand is placed to the left of the right operand.
- H **(Horizontal Cut):** Specifies that the left operand is placed below the right operand.

For example, the expression $1\,2\,V\,3\,H$ dictates that block 1 and block 2 are separated by a vertical cut (1 is left of 2), and this entire combined block is separated from block 3 by a horizontal cut (1 and 2 are below 3) (Figure 2.2).

2.2.3 Sequence Pairs

To represent the broader, more flexible space of non-slicing floorplans, the Sequence Pair representation is very commonly used [5]. A sequence pair models the relative geometric positions of all blocks using two permutations of the block set, denoted as (Γ^+, Γ^-).

For any two distinct modules a and b, their relative spatial positioning is uniquely decoded from their order in the two sequences based on the following topological rules:

- **Left/Right Relation:** If module a appears before module b in both Γ^+ and Γ^-, then a is positioned to the left of b.

$$(\dots a \dots b \dots,\ \dots a \dots b \dots) \implies a \text{ is to the left of } b \tag{2.1}$$

- **Above/Below Relation:** If module a appears before module b in Γ^+ but after b in Γ^-, then a is positioned above b.

$$(\dots a \dots b \dots,\ \dots b \dots a \dots) \implies a \text{ is above } b \tag{2.2}$$

A sequence pair can be evaluated and transformed into physical coordinates (x, y) by constructing a pair of Directed Acyclic Graphs (DAGs): the Horizontal Constraint Graph (HCG) and the Vertical Constraint Graph (VCG). The coordinates are computed by finding the longest path in these graphs, which can be executed in $O(n^2)$ time, or optimized down to $O(n \log \log n)$ or $O(n \log n)$ using Longest Common Subsequence (LCS) templates.

For the non-slicing floorplan in Figure 2.1, the sequence pairs would be:

$$\Gamma^+ = (1, 3, 4, 5, 2, 6), \quad \Gamma^- = (3, 5, 4, 1, 6, 2) \tag{2.3}$$

and the HCG and VCG would look like:

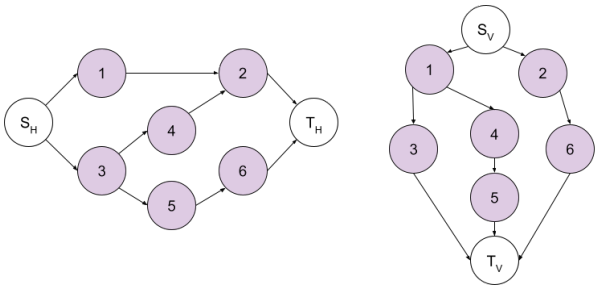


Figure 2.3: HCG and VCG for non-slicing floorplan example in Figure 2.1.

In modern deep sub-micron design, floorplanning has shifted from a pure area-packing optimization problem to a complex multi-variable constraint problem [6, 7]. Modern frameworks must also handle:

- **Hard vs. Soft Macros:** Hard macros have fixed physical dimensions and pin layouts, whereas soft macros have a fixed area but a flexible aspect ratio, allowing the optimization engine to alter their shapes dynamically to ease routing congestion.
- **Dead Space and White Space Management:** While minimizing empty, un-routable silicon area (dead space) is vital for cost, some amount of unallocated area (white space) must be preserved across the die for later standard cell placement, power grid distribution, and decoupling capacitor insertion.
- **Connectivity and Routability Overheads:** Blocks with high interconnect densities must be placed adjacent to one another to mitigate long global wire delays, changing the primary metric of floorplanning from being about optimizing area to focus more on wirelength and congestion.

2.3 Netlist Representation as Hypergraphs

To optimize a circuit algorithmically, its electrical netlist must be modeled as a mathematical structure. Traditional graph models, where an edge connects exactly two vertices $e = (u, v)$, are insufficient for VLSI circuits, as a single electrical net often connects three or more components. A hypergraph is a generalization of graphs, where hyperedges (also called nets) can connect more than two vertices. A sample hypergraph is shown in Figure 2.4.

Following the definitions used in [8], netlists are naturally modeled as a Hypergraph $H = (V, E, c, \omega)$ where:

- V is the set of vertices, representing the physical components (macros, I/O pins, and standard cells).
- E is the set of hyperedges, where each hyperedge $e \in E$ is a subset of V such that $|e| \geq 2$.
- $c : V \to \mathbb{R}_{>0}$ represents the weights assigned to vertices, typically corresponding to physical area.
- $\omega : E \to \mathbb{R}_{>0}$ represents the weights of hyperedges, typically representing connection criticality.

2.3.1 Hypergraph Partitioning

Similar to hypergraphs being a generalization of graphs, the hypergraph partitioning problem is a generalization of graph partitioning. To manage complexity, massive hypergraphs are often divided into k disjoint subsets (partitions) such that $V = V_1 \cup V_2 \cdots \cup V_k$, $V_i \neq \emptyset$ for $1 \leq i \leq k$, and $V_i \cap V_j = \emptyset$ for $i \neq j$. We then call a k-way partition Π ϵ-balanced if $\forall V_i \in \Pi$, $c(V_i) \leq L_{\max} := (1 + \epsilon)\lceil \frac{c(V)}{k} \rceil$ for some imbalance parameter ϵ.

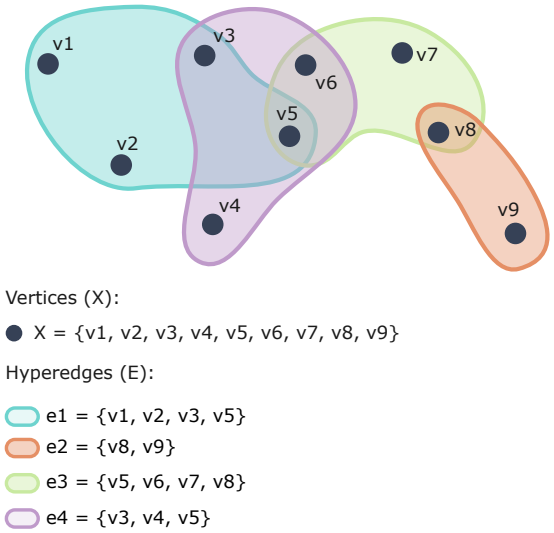


Figure 2.4: A sample hypergraph with 9 vertices, 4 nets, and 14 pins.

The k-way partitioning problem then is to find an ϵ-balanced k-way partition Π of a given hypergraph that minimizes an objective function over the cut nets. This objective is then used as the quality of a partition, and measures how effectively the found partition preserves internal logic while minimizing external communication. The objective function is commonly evaluated via three metrics:

- **Edge-Cut:** The simplest metric, counting the absolute number of hyperedges that span more than one partition.

$$\text{Cut}(H) = \sum_{e \in E} \psi(e), \quad \text{where } \psi(e) = \begin{cases} 1 & \text{if } e \text{ spans } > 1 \text{ partition} \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

- **The $(\lambda - 1)$ Metric (KM-1):** A routing-aware metric where λ_e is the number of distinct partitions connected by hyperedge e. The total cost is $\sum_{e \in E}(\lambda_e - 1)$. This severely penalizes nets that fragment across many blocks.

$$\text{Metric}_{\lambda-1} = \sum_{e \in E} (\lambda_e - 1) \tag{2.5}$$

- **Sum of External Degrees (SOED):** Defined as $\sum_{e \in E, \lambda_e > 1} \lambda_e$, this metric captures the total I/O terminal overhead required to route fragmented nets between blocks.

$$\text{SOED} = \sum_{e \in E, \lambda_e > 1} \lambda_e \tag{2.6}$$

2.4 Wirelength Estimation Models

During floorplanning and placement, the actual routing topology of the nets is unknown. The industry-standard proxy for estimating interconnect length is the **Half-Perimeter Wirelength (HPWL)**, defined as the half-perimeter of the smallest bounding box enclosing all pins of a net e:

$$\text{HPWL}(e) = \left(\max_{i \in e} x_i - \min_{i \in e} x_i \right) + \left(\max_{i \in e} y_i - \min_{i \in e} y_i \right) \tag{2.7}$$

2.4.1 Differentiable Approximations

Analytical placers utilize gradient descent optimization, which requires all objective functions to be smooth and continuously differentiable. Because the max and min operators in the HPWL equation are non-differentiable, they must be substituted with smooth mathematical approximations. Common models include the **Log-Sum-Exponential (LSE)** model:

$$\max(x_1, \ldots, x_n) \approx \gamma \ln \left(\sum_{i=1}^{n} e^{x_i/\gamma} \right) \tag{2.8}$$

and the **Weighted-Average (WA)** model, which is sometimes preferred in modern solvers to avoid the numerical overflow issues inherent to calculating large exponentials on floating-point architectures.

2.5 Analytical Global Placement

Modern global placement is formulated as a continuous, unconstrained nonlinear optimization problem. The goal is to find the physical coordinates $(\mathbf{x}, \mathbf{y})$ for all movable cells that minimize total wirelength, subject to the constraint that cells cannot physically overlap:

$$\min_{\mathbf{x},\mathbf{y}} W(\mathbf{x}, \mathbf{y}) + \lambda D(\mathbf{x}, \mathbf{y}) \tag{2.9}$$

where $W(\mathbf{x}, \mathbf{y})$ is the approximated wirelength, $D(\mathbf{x}, \mathbf{y})$ is the density penalty function, and λ is a penalty multiplier that dynamically increases as the optimization progresses.

2.5.1 The Electrostatic Density Analogy

To calculate the density penalty $D(\mathbf{x}, \mathbf{y})$ efficiently across millions of cells, some tools employ an electrostatic analogy (ePlace). The layout area is discretized into a grid, and cells are modeled as positively charged particles. The density gradient is derived by computing the electrostatic potential field $\Phi(x, y)$, which requires solving Poisson's equation:

$$\nabla^2 \Phi(x, y) = -\rho(x, y) \tag{2.10}$$

where $\rho(x, y)$ is the spatial charge density. This equation is efficiently solved in the spectral domain using Fast Fourier Transforms (FFTs) in $O(N \log N)$ time, creating a repulsive vector field that pushes cells away from highly congested regions.

2.5.2 Nesterov's Accelerated Gradient

The resulting optimization surface is highly non-convex. To navigate this, analytical placers rely on first-order optimization algorithms such as Nesterov's Accelerated Gradient (NAG). Unlike standard gradient descent, NAG incorporates a "momentum" term that relies on historical coordinate updates, allowing the solver to intelligently accelerate through flat regions of the cost surface and converge more reliably on high-quality local minima.

Chapter 3

State of the Art

While traditional industrial floorplanning solutions that only focus on fixing the position of macros are fast, they often produce inadequate solutions that require a lot of iteration. This has led to many advancements in the optimization of placement of mixed-size circuits. The methodology of mixed-size circuit placement has undergone significant evolutionary shifts as design complexities have scaled into billions of gates. This chapter contextualizes the proposed clustering and floorplanning framework by reviewing the state of the art across three core domains: hypergraph partitioning, floorplanning & mixed-size placement, and industry standards.

3.1 Hypergraph Partitioning and Clustering

To mitigate the high dimensionality of flat placement on modern designs, logical clustering has been long studied as a pre-processing step. Circuits are naturally modeled as hypergraphs, making hypergraph partitioning a foundational optimization problem in electronic design automation (EDA). In [9] they state two different problems. One is clustering for delay minimization with replication (CA), meaning that individual circuit components may be assigned to more than one cluster. While CA has been shown to be solvable in polynomial time [10], unbounded replication can be quite expensive. Instead solving without replication (CN), where each component belongs to one cluster only, has grown as a relevant subject of study in VLSI. Specifically in [9] the authors establish a modeling framework for logic gates using directed acyclic graphs (DAGs), and prove the NP-hard nature of many variants of CN.

Recently new hypergraph partitioning packages like `hMETIS` [11] have emerged as the industry standard. These algorithms operate by recursively coarsening the hypergraph, partitioning the coarsest representation, and then uncoarsening it while applying local refinement heuristics. The authors in [8], for example, combine some pre-processing strategies like Louvain community detection to guide the coarsening step, and then do a highly localized local search while re-

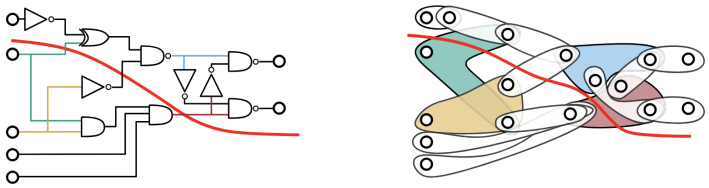


Figure 3.1: A circuit (left) represented as a hypergraph (right) with a two-way partition cutting three nets. Taken from [8].

versing the contraction process, outperforming many other graph partitioning tools. They provide this partitioner as an open-source tool called KaHyPar .

Other newer research looks into partitioning hypergraphs using spectral algorithms rather than multilevel coarsening, in order to avoid entrapment in local minima, and to avoid over relying on the community properties of graphs [12]. In [13] they propose a new type of hypergraph learning for neural networks, which they call Hypergraph Neural Networks (HNNs), on top of which new research is building to develop alternative partitioning strategies [14].

3.2 Floorplanning & Mixed-Size Placement

Floorplanning represents a critical phase in the physical design hierarchy, responsible for determining the rough spatial arrangement of the larger functional blocks. Historically, floorplanning methodologies are categorized into slicing and non-slicing floorplans. Slicing floorplans can be recursively divided into smaller rectangles using horizontal or vertical cutting lines, a property that simplifies the optimization space [4, 15]. Instead non-slicing floorplans, which cannot be divided through cutting lines, allow for more complex geometric arrangements but require more sophisticated representations to avoid overlaps. To navigate this design space, various topological data structures have been proposed to represent relative positions without fixing exact coordinates, such as B*-trees [16], Sequence Pairs [5], and O-trees [17]. By the mid-1990s, traditional SA-based placers [18], which relied on random perturbations of these data structures, struggled to scale. This led to the adoption of deterministic partitioning-based methods like the ones based on min-cut algorithms [19, 20] and tree/graph representations [17] to handle structural complexity.

As modern integrated circuits integrate billions of transistors, mixed-size placement has also emerged as a dominant challenge. This paradigm requires algorithms to handle electronic components that are radically different in size, ranging from massive IP blocks (macros) to millions of minuscule standard cells. Historically, tools decoupled this problem by doing a floorplan with macros first and standard cells second. Tools like the one presented in [21] focus on fixing large blocks by pushing and anchoring them strictly along the chip boundaries

to lock in the core topology before standard cell initialization. Yet some current research shows the importance of placing macros and standard cells roughly at the same time (concurrent or flat global placement) rather than in separate phases [22, 23]. This simultaneous approach allows the solver to find a globally optimal equilibrium between macro blockages and standard cell routing demands. Analytical placement has re-emerged as the dominant paradigm. Modern flat global placers, like RePlAce [24] (available in the OpenROAD project [25]), initially optimize the design holistically. They allow temporary, illegal overlaps across mixed-size components before using Nesterov's method to smoothly distribute density. In RePlAce, following the ideas of [23], the overlap-prevention constraint is modeled as an electrostatic field.

Furthermore, handling components of vast scale differences requires utilizing structural metrics and data-affinity. Hierarchical and multilevel placement strategies are another big part of modern VLSI research [26–30]. They manage complexity by clustering netlists based on dataflow awareness, the timing constraints, or data-affinity of connections. By analyzing the position of I/O pins and resistor-transistor logic (RTL) connectivity, they ensure standard cells are grouped naturally near the memories and register boundaries they serve. These architectures often employ metaheuristics, like Simulated Evolution [31], to iteratively evolve the placement under strict aspect-ratio and signal-integrity constraints.

More recently, the current research frontier increasingly incorporates artificial intelligence to manage these mixed-size dynamics, particularly by intelligently clustering standard cells into local blocks before or during macro alignment. Reinforcement Learning (RL) has been applied to learn spatial heuristics, placing macros dynamically based on concurrent standard cell density [32], and in routability models like RUDY [33]. Beyond RL, collaborative frameworks such as CHIP-MAP [34] integrate Large Language Models (LLMs) to orchestrate macro placement. By combining semantic design intent with analytical solvers, these AI-driven frameworks predict how standard cells will cluster around specific functional logic, allowing the tool to co-optimize macro and standard-cell boundaries more effectively. To overcome the massive computational bottleneck of these concurrent, mixed-size formulations, recent breakthroughs cast the flat analytical placement problem as a neural network training problem. DREAMPlace [35] replaces standard deep learning weight updates with coordinate optimizations. By leveraging GPU-accelerated toolkits like PyTorch, it solves multi-million gate, mixed-size placements in minutes instead of hours, bridging the gap between massive macro blocks and standard cells seamlessly.

3.3 Industry Standards

To validate new algorithms, the EDA community relies on standardized benchmark suites and flexible translation tools that accurately mimic industrial complexities. Commonly used benchmarks for global placement and floorplanning include those of the ISPD 98, ISPD 2005, ISPD 2006, ICCAD 2012, and IC-

CAD 2015 [36]. These benchmarks are created as challenges derived from real industrial ASIC designs, and present diverse physical layouts with high density, multiple macro configurations, and tight timing walls. Some of them present all I/O pins in the corners, while others have area-array I/Os mixed in between the macros.

A persistent challenge in both academic and industrial research is the fragmentation of physical design formats. While academic research often utilizes simplified, abstract representations (such as `.nodes`, `.nets`, and `.wts` [37]), industry production pipelines rely on detailed geometric and logical layouts (such as `LEF/DEF` or OpenDB databases [38]). Infrastructure projects like RosettaStone [39] bridge this structural gap, providing conversion utilities that ensure academic algorithms can be directly evaluated on production-ready, industrial-grade data formats. These translation and standardization are critical to proving the viability of proposed hierarchical frameworks on configurable computing systems and modern SoCs [40]. Other research has also focused on the creation of "horizontal benchmarks", to try to share one common format across all design stages, and enable maximal comprehension across academic and commercial tools [41].

Chapter 4

Methodology

The methodology is structured into 6 stages shown in Figure 4.1: benchmarks pre-processing, netlist hypergraph modeling, multi-way partitioning with metric extraction, floorplanning with block partitions, macros placement and fine-tuning, and global standard cells placement. All phases also involve changes in the formats and files in which the netlists and hypergraphs are represented.



Figure 4.1: Flow diagram of the methodology followed.

4.1 Benchmarks pre-processing

Before building the hypergraphs, a preliminary analysis of all the benchmarks publicly available was done. Many of these were not designed for mixed-size placement, and the ones that were (ISPD02, ICCAD04) are no longer representative of modern VLSI design. For this reason, the ISPD 2005 and 2006 benchmarks were used with a few modifications, following similar steps as the ones taken by [28] to turn them into actual mixed-size placement problems. The transformations used were:

1. For benchmarks with previous fixed macro and I/O pin positions, all macros where freed. Then to classify between macros and terminals, two strategies were tested:

    - Following [28], a λ value was considered such that the initially fixed components bigger than λ times the average size of all components of the design were considered macros, and the rest were considered I/O pins.

- Using k-means clustering all the components were separated into 3 clusters; one corresponding to standard cells, one for I/O terminals, and one for macros.

A few special cases where also considered:

- *adaptec1*, *adaptec2*, *bigblue1*, *newblue1* have all their terminals in the perimeter of the design. For these cases, the die was first reduced into its core area. All components in the removed borders were directly considered terminals regardless of size, and were displaced vertically or horizontally to be handled as points on the edges. An example of this pin displacement is shown in figure 4.2.
- *bigblue2* and *bigblue4* have a single isolated macro not connected to anything. These were treated as fixed objects so that they would not affect the floorplanning.

2. In early experimental iterations, I/O terminals and peripheral pins were modeled as full physical objects with non-zero dimensions. This design choice led to many failures during the global floorplanning and legalization stages. Some of the circuits within the benchmarks contain thousands of I/O terminals, treating them as solid physical blocks caused severe spatial congestion, which then caused the legalization process of the floorplanner to be over-constrained resulting in unfeasible problem formulations. To resolve this issue, again following the steps in [28], all I/O terminals' size was set to zero. In modern designs, area-array I/Os are usually in separate metal layers and are allowed to overlap with other movable objects. This means that removing their sizes is still representative of contemporary industrial designs.

3. A temporary netlist was produced with all the terminals removed, as they are not considered for the hypergraph partitions.

4. After analyzing net size and disparity for each circuit (shown in Figure 4.3), large nets connecting 100 or more components were removed as they are not meaningful when doing the graph partitioning, but increase a lot the memory used for exploration and the overall processing time.

Before partitioning, when classifying the circuits' modules between Standard Cells, I/O pads, and Macros using the λ factor generally resulted in more challenging problems (more macros, as seen in Table 4.2). It also produced more I/O terminals, which then serve as reference points so the floorplanner gets a better initial framework for the layout. The only exception was with *newblue3* where, due to the distribution of component sizes (roughly 18 enormous macros, 33 small macros, and thousands of large terminals), using an unsupervised k-means classification was heavily biased towards the smaller components and resulted in 678 macros. Circuits were then processed using the idea that solving

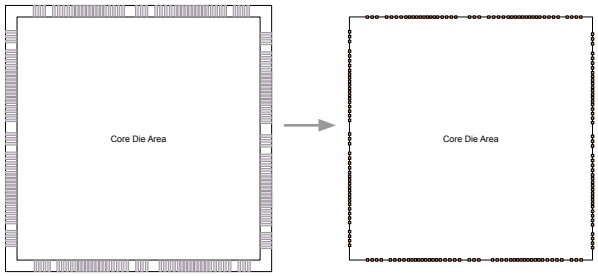


Figure 4.2: Sample diagram showing how the pins in the border regions were displaced, and the die reduced to its core area.

the more complex problem produces also a valid solution for the smaller problem. Only when one of the configurations failed, the other classification method was tested to attempt to get results.

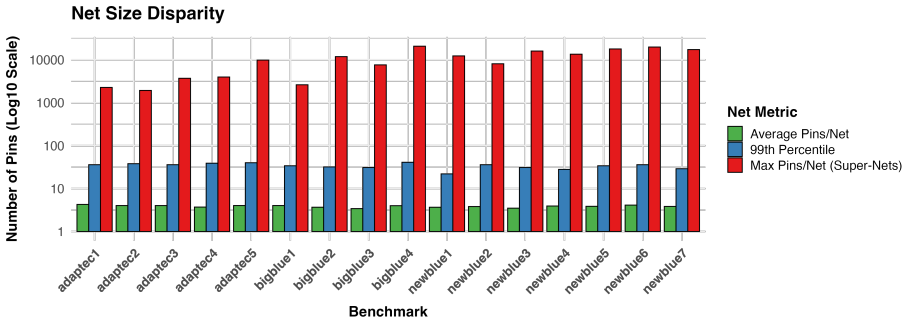


Figure 4.3: Distribution of hyperedege (nets) sizes for each benchmark. Most circuits have a lot of small nets, and a few very large ones.

For this first step the input benchmarks had the standard Bookshelf format with `.nodes`, `.nets`, `.pl`, and `.scl` files; as described in [37, 42]. General metrics for the benchmarks used after the pre-processing step when using both classification strategies are shown in tables 4.1 and .

4.2 Netlist Hypergraph Modeling

As discussed in Section 2.3, to represent the complex connectivity of modern VLSI circuits, the input netlist benchmarks are modeled as a hypergraph $H = (V, E, c, \omega)$.

One challenge faced was the extreme disparity in physical area between massive macro blocks and tiny standard cells. To prevent the partitioning engine from being biased by these dimensions, a weight normalization strategy was

Benchmark	**#Modules**	**#Nets**	**#Movable**	**Target Density** (%)
adaptec1	211447	221142	210904	100
adaptec2	255023	266009	254457	100
adaptec3	451650	466758	450927	100
adaptec4	496045	515951	494716	100
bigblue1	278164	284479	277604	100
bigblue2	557866	577235	534782	100
bigblue3	1096812	1123170	1095519	100
bigblue4	2177353	2229886	2169183	100
adaptec5	843128	867798	842482	50
newblue1	330474	228901	330137	80
newblue2	441516	465219	330239	90
newblue3	494011	552199	482833	80
newblue4	646139	637051	642717	50
newblue5	1233058	1284251	1228177	50
newblue6	1255039	1288443	1248150	80
newblue7	2507954	2636820	2481372	80

Table 4.1: General statistics and design characteristics of used Benchmarks.

Benchmark	**#Modules**	λ	**#Macros** (λ)	**#Fixed I/O** (λ)	**#Macros (k-means)**	**#Fixed I/O (k-means)**
adaptec1	211447	70	63	480	49	494
adaptec2	255023	160	127	439	28	538
adaptec3	451650	650	58	665	47	676
adaptec4	496045	460	69	1260	61	1268
bigblue1	278164	120	32	528	23	537
bigblue2	557866	30	960	22124	925	5528
bigblue3	1096812	470	65	1233	55	615
bigblue4	2177353	550	200	7970	158	497
adaptec5	843128	440	76	570	46	600
newblue1	330474	100	64	337	16	385
newblue2	441516	190	25	1252	8	1193
newblue3	494011	170	51	11127	678	3490
newblue4	646139	400	81	3341	77	506
newblue5	1233058	570	91	4790	426	692
newblue6	1255039	650	74	6815	60	1270
newblue7	2507954	650	161	26421	58	1843

Table 4.2: Components classification of used Benchmarks using λ factor and k-means.

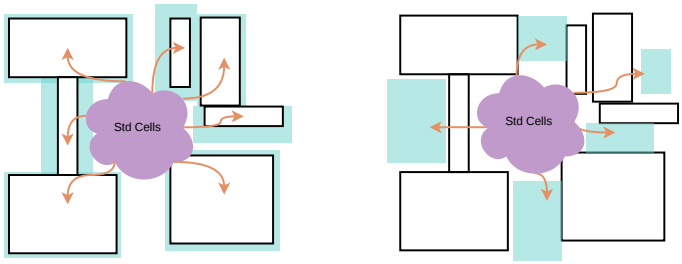


Figure 4.4: Different strategies explored for the hypergraph partitioning, and their meaning in terms of standard cells distribution. To the left, blue areas represent the added size to each macro. To the right, blue areas represent newly created blocks, that do not contain any macros.

applied, where every vertex belonging to a macro was assigned a uniform weight:

$$c(v_i) = 1, \quad \forall v_i \in M, \tag{4.1}$$

where M is the subset of V containing only the macros.

This strategy ensures that the partitioning algorithm optimizes solely based on netlist topology and logic clusters, preventing area-heavy macros from prematurely constraining the initial logic grouping.

For this phase, the file format used was the one described in `hMETIS` [11], with files `.hgr` and `.fix`. This format is common across hypergraph partitioners, and one of the accepted formats for the `KaHyPar` python package used.

4.3 Partitioning and Metric Extraction

The modeled hypergraph is subjected to multi-way partitioning to divide the netlist into $m + k$ disjoint partitions, where m represents the number of macros in the benchmark. The design space is then explored by varying two primary parameters: the number of target standard-cell only partitions (k) and the maximum allowable area imbalance (ϵ).

Generally speaking, the value of k serves two testing purposes:

- When $k = 0$, all the standard cells are "attached" to their respective macros with the biggest data-affinity to create balanced blocks.
- As k becomes larger, more standard cells are separated from the macros, and distributed into newly created clusters with the other standard cells that belong to similar nets.

This effect is shown in figure 4.4.

To evaluate the quality of each partitioning solution, we extract three standard connectivity metrics that serve as independent variables in our correlation

analysis. Regardless, the partitioning is done with a configuration to try to optimize for the smallest KM-1 metric. This was done using the multi-thread python interface of the KaHyPar partitioner [11]. An simplified example of a graph that represents the partitioning of a hypergraph is shown in Figure 4.5.

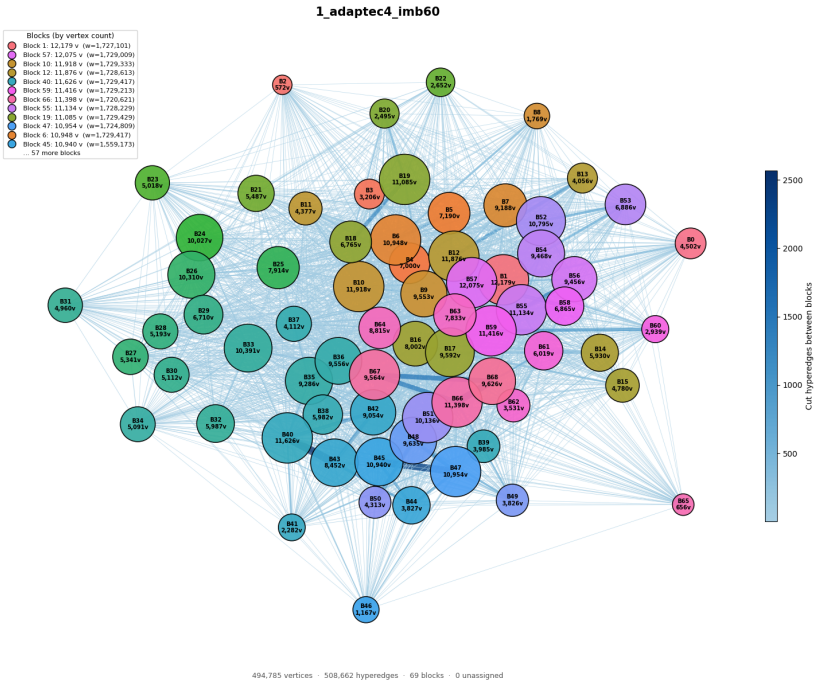


Figure 4.5: Example of a partitioned hypergraph generated from the *adaptec4* benchmark using an imbalance of 60%. Each vertex corresponds to a block partition and the edges in the visualization represent the number of hyperedges cut between that pair of blocks. Thicker, darker lines represent more tightly connected blocks.

4.4 Block Reconstruction and Floorplanning

4.4.1 Block Reconstruction

Following the extraction of logical metrics, and after having a partitioned hypergraph, the partitions are transformed back into physical entities to prepare for the floorplanning stage.

Each partition B_i is reconstructed as a hybrid block. This block is composed of an optional single Macro, and a Standard Cell Cluster. The total area of a

hybrid block $A(B_i)$ is calculated as:

$$A(B_i) = \left(A_{macro} + \sum_{v \in P_i} A_{cell}(v) \right) \times (1 + \alpha) \tag{4.2}$$

where α represents a white-space padding factor used to improve routing congestion during the next design stages.

This area is then used to model the blocks according to a range of target aspect ratios $[AR_{min}, AR_{max}]$, to provide the floorplanner with more flexibility for area optimization. These values are calculated as:

$$AR_{min} = \frac{\max_{v \in P_i}(width(v))^2}{A(B_i)} \tag{4.3}$$

$$AR_{max} = \frac{A(B_i)}{\max_{v \in P_i}(height(v))^2} \tag{4.4}$$

4.4.2 Floorplanning

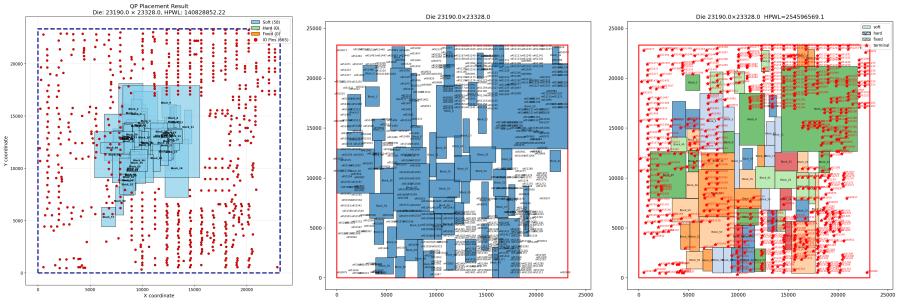


Figure 4.6: Sample of the multi-stage floorplanning process using *adaptec3*. To the left the result of quadratic placement, in the middle the global floorplanning, and to the right the result after legalization.

For this step, the multi-stage solution showed in [43] was used. As such the whole floorplanning process is divided into stages, under the main idea that a good initial placement can significantly improve the search later performed by convex and non-convex optimizers. The three stages used for finding the blocks location, shown graphically in Figure 4.6, are:

1. **Quadratic Placement:** Pre-computes the initial location of the modules by solving a sparse system of linear equations. Internally, hypergraphs are transformed into graphs by using the clique model for hyperedges.

2. **Global Floorplanning:** Refines the initial locations by performing a reshuffling-expansion algorithm, while satisfying the area and aspect-ratios and trying to reduce the HPWL. In the re-shuffling phase, the centroids of

objects are swapped and evaluated using standard Simulated Annealing (SA). Then, in the expansion phase, objects are iteratively repelled and reshaped until the overlap fails to improve beyond a small tolerance.

3. **Legalizing:** The global floorplanning phase provides good preliminary positions for all the modules, but it still presents some overlap. To remove this overlap and legalize the positions, a hybrid-space convex model with LSE approximation is built from the Horizontal Constraint Graph (HCG) and the Vertical Constraint Graph (VCG), which guarantees global optimality for a given topology.

For this phase the sizes, placements, and definitions from the original Bookshelf files had to be merged with the results of the hypergraph partitioning files, and put into a netlist file following the Floorplanning Exchange Format (FPEF), and Die Exchange format (DIEF) described in [44].

4.5 Macros Placement

Once the position of the reconstructed blocks B_i is fixed and legalized, and before starting the standard cell global placement, the compound blocks must be dissolved. This breaks down each B_i into its parts: the single physical macro M_i and the unplaced standard cell cluster P_i.

Because each hybrid block is dimensioned with whitespace padding (α) and a surrounding sea of standard cells, the macro M_i typically uses only a fraction of the block's area $A(B_i)$. This means there is a bounded range of movement inside the block's coordinates where the macro can be placed. The exact position of M_i within its block B_i also has an impact on the wirelength, timing, and routability of both its own pins and the surrounding standard cells. The effect of positioning the macros within the blocks is shown in Figure 4.7.

As a baseline, macros are initially positioned at the center of their legalized blocks:

$$x_{M_i} = x_{B_i}^{\min} + \frac{W(B_i) - W(M_i)}{2}, \quad y_{M_i} = y_{B_i}^{\min} + \frac{H(B_i) - H(M_i)}{2} \tag{4.5}$$

where $(x_{B_i}^{\min}, y_{B_i}^{\min})$ is the lower-left corner of block B_i, and $W(B_i)$, $H(B_i)$, $W(M_i)$, $H(M_i)$ correspond to the widths and heights of the block and macro, respectively.

However, placing the macro in the center is rarely optimal. To minimize the local interconnect length, a Pin-Affinity and Cell-Centroid Shifting (PACCS) strategy was implemented, following some ideas used by [45, 46]. The displacement of a macro from the center would be guided by two competing, pull-like forces:

1. **The Local Clustered Standard Cell Pull:** The logical standard cells in the cluster P_i have a physical center of mass (centroid) $C(P_i) = (x_c, y_c)$

which can be defined as:

$$x_c = \frac{1}{|P_i|} \sum_{v \in P_i} x(v), \quad y_c = \frac{1}{|P_i|} \sum_{v \in P_i} y(v) \tag{4.6}$$

where $(x(v), y(v))$ is the estimated physical coordinate of standard cell v. Moving the macro closer to this centroid shortens the average length of standard cell-to-macro connections. For the specific implementation used here though, as there is no previous estimated physical coordinate for each standard cell, we assume a uniform distribution across the additional white-space. This results in $C(P_i)$ simply reducing into the block's center, and the Standard Cell Pull acting as a "centering" force.

$$C(P_i) \approx \left(x_{B_i}^{\min} + \frac{W(B_i)}{2},\ y_{B_i}^{\min} + \frac{H(B_i)}{2} \right) \tag{4.7}$$

2. **The External Interconnect and Pin Access Pull:** External signals connecting to the macro's physical pins are routed over the block's boundary. For each pin $p_j \in \text{Pins}(M_i)$, let $\vec{r}_{p,j}$ be its relative offset from the macro's center and $\vec{x}(\text{target}_j)$ be the absolute coordinate of its target pin outside B_i.

Taking these forces into account, we calculate a target coordinate $\vec{x}^* = (x_{M_i}^*, y_{M_i}^*)$ for the macro's center of mass using a weighted affinity vector:

$$\vec{x}^* = \frac{\gamma_c C(P_i) + \sum_j \omega_j \left[\vec{x}(\text{target}_j) - \vec{r}_{p,j} \right]}{\gamma_c + \sum_j \omega_j} \tag{4.8}$$

where γ_c represents the local clustering affinity factor, and ω_j represents the net weight of the connection associated with pin p_j. For this implementation, ω_j was defined as:

$$\omega_j = \frac{1}{D - 1} \tag{4.9}$$

where D is the net degree. Using a fixed value of γ_c would not make sense when balancing macros with different amounts of pins, so instead it was calculated with a β factor as:

$$\gamma_c = \beta \cdot W_{\text{pins}} \cdot \left(\frac{|P_i|}{\text{Pins}(M_i)} \right) \tag{4.10}$$

where $|P_i|$ is the number of standard cells assigned to the block, $\text{Pins}(M_i)$ is the number of pins in macro i, and W_{pins} is the total topological weight of all pin pulls $\sum_j \omega_j$.

Finally, to ensure the macro remains in a legal position and does not exceed the bounding box of its block B_i, the final displacement is projected onto the valid container with a clearance margin δ from the block's boundary:

$$x_{M_i} = \max\left(x_{B_i}^{\min} + \delta, \min\left(x_{M_i}^*, x_{B_i}^{\max} - W(M_i) - \delta \right) \right) \tag{4.11}$$

$$y_{M_i} = \max\left(y_{B_i}^{\min} + \delta, \min\left(y_{M_i}^*, y_{B_i}^{\max} - H(M_i) - \delta \right) \right) \tag{4.12}$$



Figure 4.7: Example of macros placement in the *adaptec1* benchmark. To the left, placement of blocks after floorplanning in the reduced core die. To the right, placement of macros at the center of their blocks, in the fully sized die.

4.6 Global Placement

With the macros shifted, legalized, and completely fixed in their final locations, global placement of the free standard cells is performed. For this phase, we utilize an analytical global placement engine driven by Nesterov's optimization method, based on the RePlAce algorithm [24] and integrated directly within the OpenROAD flow.

In the OpenROAD framework, the global placement step is invoked via the `global_placement` command. RePlAce treats the placement domain as a continuous optimization problem. The core objective is to place standard cells inside the core area while satisfying two conflicting requirements: minimizing overall interconnect length and eliminating physical cell overlaps (density spreading).

Mathematically, this is modeled as an unconstrained non-linear optimization problem:

$$\min_{\mathbf{x},\mathbf{y}} F(\mathbf{x},\mathbf{y}) = W(\mathbf{x},\mathbf{y}) + \lambda D(\mathbf{x},\mathbf{y}) \tag{4.13}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the coordinate vectors of the movable standard cells, $W(\mathbf{x},\mathbf{y})$ is the total wirelength of the design, $D(\mathbf{x},\mathbf{y})$ is the overlap/density penalty function, and λ is a dynamic scaling parameter.

To model the cell density penalty $D(\mathbf{x},\mathbf{y})$, RePlAce uses a novel electrostatic analogy. The entire placement area is divided into a uniform grid of 2D bins; standard cells are modeled as positively charged particles, where each cell's charge is equal to its physical area; and the density penalty is modeled as the total electrostatic potential energy of the system.

To make this process computationally viable for modern designs with millions of cells, OpenROAD utilizes a 2D Fast Fourier Transform (FFT) on the grid

bins, solving Poisson's equation in the spectral domain in $O(N \log N)$ time. The optimization is solved using Nesterov's Accelerated Gradient (NAG) descent, which is a first-order optimization method with momentum. The dynamic multiplier λ starts at a small value to allow wirelength-driven optimization, and is progressively scaled up based on density convergence metrics to drive cell spreading until the global placement is free of major overlaps and ready for local legalization.

This phase required translating the FPEF format used previously by the floorplanner, into the industry standard LEF/DEF format [38].

4.7 HPWL Measurement

The metric of choice to evaluate and optimize wirelength throughout the floorplanning and global placement stages is the Half-Perimeter Wirelength (HPWL). HPWL is the industry-standard proxy for estimating interconnect routing length during early physical design before routing is actually run.

For a net $e \in E$ connecting a set of pins V_e, let the physical location of pin $i \in V_e$ be denoted as (x_i, y_i). The HPWL of net e is defined as the half-perimeter of the bounding box enclosing all pins of the net:

$$\text{HPWL}(e) = \left(\max_{i \in V_e} x_i - \min_{i \in V_e} x_i \right) + \left(\max_{i \in V_e} y_i - \min_{i \in V_e} y_i \right) \tag{4.14}$$

The total HPWL of the layout is the sum of the HPWL calculations over all nets in the netlist:

$$\text{HPWL}_{\text{total}} = \sum_{e \in E} \text{HPWL}(e) \tag{4.15}$$

For 2-pin and 3-pin nets, the HPWL is mathematically identical to the Minimum Rectilinear Steiner Minimum Tree (RSMT) length. For multi-pin nets, HPWL acts as a tight, computationally inexpensive lower bound.

Chapter 5

Results

This chapter presents a comprehensive evaluation of the proposed placement framework. The algorithm was fully implemented using Python language, and ran as two separate phases on separate computing instances. The first phase, consisting of the hypergraph partitioning and floorplanning, was run in the rdlab cluster server provided by the department of Computer Science - High Performance Computing (HPC) in the Universitat Politècnica de Catalunya (UPC), with a an Ubuntu configuration using 16GB of RAM and a 4 core CPU. The second phase, consisting of the standard cells placement using OpenROAD, was run in a Macbook Pro with an M1 Pro CPU with 16 GB of RAM.

5.1 Hypergraph Partitioning

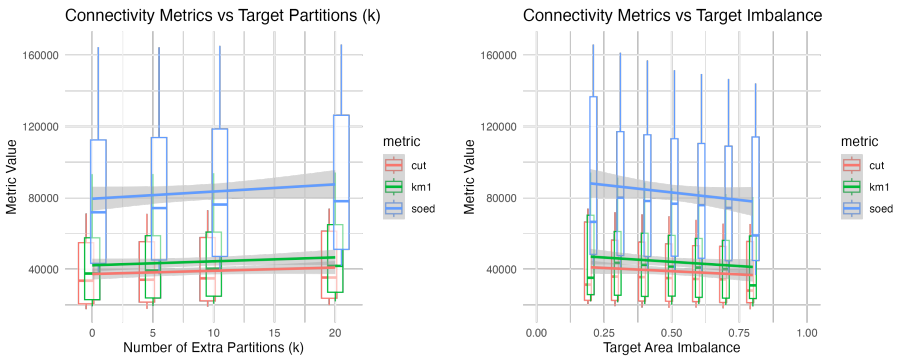


Figure 5.1: Behavior of the measured hypergraph metrics (edge-cut, KM-1, and SOED) as number of partitions (left) or imbalance (right) increase.

The results shown in Figure 5.1, when using all the generated hypergraphs and their partition metrics, showcase two noticeable trends. First is that as

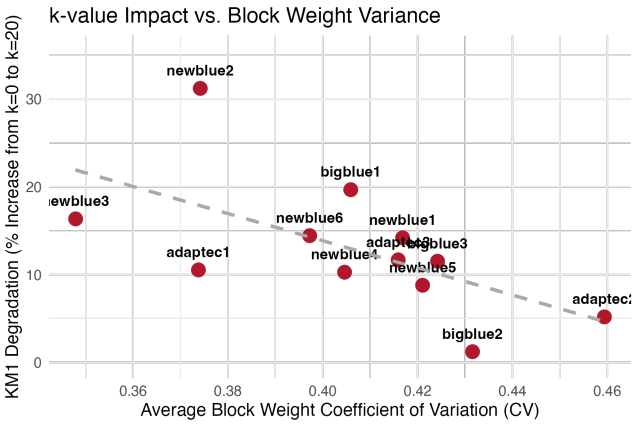


Figure 5.2: KM-1 degradation as the average block weight variation (CV) increases. Designs with higher variance tend to have a more stable connectivity after partitioning.

the number of partitions increases, the edge-cut, KM-1, and SOED metrics all increase. Second that a higher imbalance gives more freedom to the partitioner to look for configurations that minimize them, resulting in decreasing values.

Another observation comes from the correlation between the measured metrics, shown in Table 5.1. Having values so close to 1 indicates an extremely strong correlation meaning that for these benchmarks, regardless of which value is chosen as the minimization criteria during the hypergraph partitioning, the results will be nearly the same. This indicates also that all three metrics will behave roughly equally well as indicators for the design's quality measured in HPWL.

	cut	**km1**	**soed**
cut	1.0000	0.9841	0.9953
km1	0.9841	1.0000	0.9967
soed	0.9953	0.9967	1.0000

Table 5.1: Pearson Correlation Matrix for Connectivity Metrics

It is important to note, however, that not all benchmarks are equal. Some exhibit structural characteristics that can make effective floorplanning more complex later on. One of this characteristics is the distribution in sizes of its components. For instance, as seen in Figure 5.2, there is a negative correlation between the mean block weight variance (CV) and the degradation of the KM-1 when standard cells are partitioned into independent clusters ($k > 0$). This suggests that benchmarks with a naturally higher variance in block sizes actually experience less logical degradation, and circuits with more homogeneous

macros benefit from having a lower k value. Additionally, although it may seem counterintuitive, from Figure 5.3 it is observed that the block weight variance is an inherent property of the circuit's design, and not simply a byproduct of the partitioning imbalance. While enforcing a smaller target imbalance reduces CV, the data reveals that the fixed topological and physical characteristics of the circuits are far more relevant than the partitioner's area constraints.



Figure 5.3: Average block weight variance (CV) compared to average node weight (left), and CV for each design and imbalance (right). The CV is heavy dependent on the benchmarks' topology, and not exclusively dependent on the partitioning imbalance.

5.2 Floorplanning

Transitioning from logical graph partitions into physical, legalized hybrid blocks proved to be one of the most challenging steps in the overall physical design.

The global floorplanning step relies on a non-deterministic algorithm that performs reshuffling-expansion using SA. The empirical findings of this research show how sensitive the whole floorplanning is to the random nature of that step. If the output initial positions of the components were not good, it was then impossible for the legalizer to converge to a solution. This was very likely the reason why some benchmarks with less spatial constraints and a larger search space (like *bibglue2*, *bigblue3*, or *newblue1*) failed on some of the trials. Changing the seed used sometimes resulted in legal positions, but this complicated the setup of a reliable experimentation framework that could always produce valid solutions.

The case of *newblue3* was very particular, as it contains a very high density of massive macros with a very small size variation, and the floorplanner was not able to converge to any feasible solution for any of the tested parameters. The

most likely cause is that there is a very tight vertical constraint that results in, practically speaking, only one possible configuration for the largest 18 macros. Some examples of attempts with their unfixable overlaps are shown in Figure 5.4. The total number of failed floorplans for each benchmark is shown in Table 5.2.

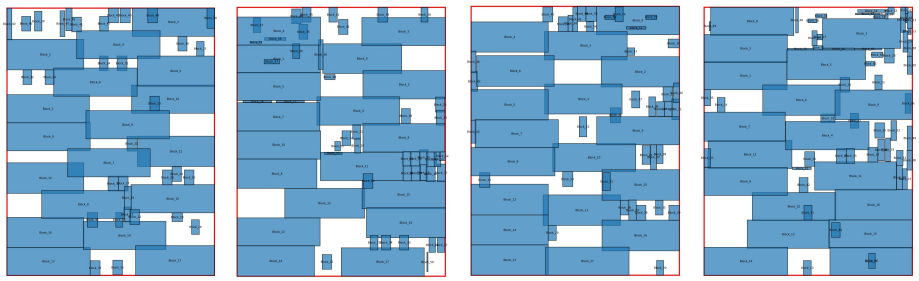


Figure 5.4: Four samples of attempting to do floorplanning for the *newblue3* benchmark. From left to right, the parameters for imbalance ϵ and number of partitions k increases.

Another observation was how the execution time scales with the number of blocks, regardless of the size and weight variance of these. Although exact times were not measured, *bigblue2* was consistently the slowest benchmark to process as it has several hundreds of small but similar-sized macros. Compared to it, even if *bigblue4* and *newblue7* are the biggest designs by number of nodes and nets, they resulted in a faster convergence.

Benchmark	**Total Floorplans**	**Successful**	**Failed**
newblue3	20	0	20
bigblue2	20	17	3
adaptec2	20	18	2
bigblue3	20	18	2
newblue1	20	18	2
newblue2	20	18	2
newblue5	20	19	1
adaptec1	20	20	0
adaptec3	20	20	0
adaptec4	20	20	0
adaptec5	20	20	0
bigblue1	20	20	0
bigblue4	20	20	0
newblue4	20	20	0
newblue6	20	20	0
newblue7	20	20	0

Table 5.2: Number of failed floorplans per benchmark.

5.3 Placement

Once the hybrid blocks are legalized, they are broken apart to expose the individual macros. This section analyzes the physical impact of the macro placement step and the standard cells placement, comparing two distinct placement strategies:

- Placing all macros at the center of the blocks.
- Using the proposed Pin-Affinity and Cell-Centroid Shifting (PACCS) algorithm to displace the macros according to the pulling forces of the pins to which each macro is connected to.

To analyze the results from the HPWL, the taken wirelength measurements were first normalized using 1 as the best found solution. Looking at Figures 5.5 when the macros are simply placed at the center we observe two interesting characteristics. As predicted from $\mathbf{H}_{2a}$, the relationship between k and the HPWL is non-monotonic, and on average the quality of the designs were best when $k = 5$, and $k = 10$. From $k = 20$ and onward, the quality seems to worsen, and the partitioner increasingly produces empty blocks. Then, we also observe that even though $\mathbf{H}_{2b}$ was also correct and a higher imbalance ϵ gives more freedom to the partitioner to prioritize the topology, the impact is seemingly not as strong.

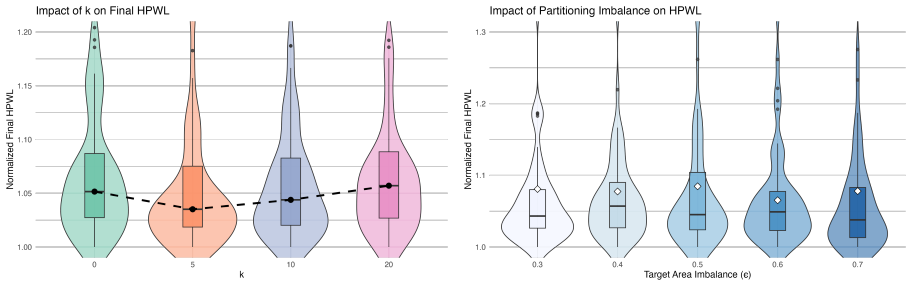


Figure 5.5: Violin plots showing the full distribution of normalized HPWL for different values of k (left) and ϵ (right).

After observing that the partitioning metrics were very highly correlated in section 5.1, it was expected that they would also be roughly equally correlated with the final HPWL. Nevertheless, even if none of the metrics was a reliable indicator of the performance of any of the benchmarks (Figure 5.6), when looking at the overall normalized values across all benchmarks (Table 5.3), it was clear that there is a slightly stronger association between the design's quality and KM-1, followed by SOED, and lastly Edge-Cut.

Comparatively, when the macros where displaced from the center of the blocks using the PACCS implementation, the best found solutions performed a 0.58% worse. While the difference is small, on average all the PACCS solutions

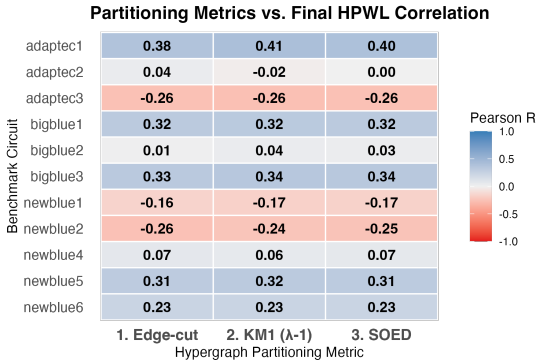


Figure 5.6: Correlation matrics of partitioning metrics with HPWL for each design. We observe all metrics behave roughly the same, and none of them is a clear indicator of wirelength.

	$\lambda - 1_{\text{norm}}$	$\textbf{SOED}_{\text{norm}}$	$\textbf{Edge-Cut}_{\text{norm}}$
$\textbf{HPWL}_{\text{norm}}$	0.1238	0.1022	0.0779

Table 5.3: Average correlation of normalized Hypergraph Metrics with the final normalized HPWL for all benchmarks.

resulted in 10% higher HPWL than the best for each benchmark, compared to only an 8% with the centered macros (Figure 5.7). These results indicate that the optimal solution likely relies more heavily on having the macros as close as possible to the standard cells with the highest data affinity and less on moving towards the the external pins that it is connected to (a higher β parameter). Having a better defined centroid $C(P_i)$ for the standard cells within the blocks, and quantifying exactly how much does the macro displacement affect the wirelength would be an interesting future area of study.

As also seen in Figure 5.7, there was an interesting outlier with a normalized HPWL of 2.48. This point corresponds to *newblue2*, when using an imbalance of $\epsilon = 0.5$ and a $k = 5$. From the OpenROAD output logs of the placement process it was observed that the explosion of HPWL only occurred in the last few iterations. The most likely reason why this occurred is because of a lack of necessary padding and white-space between macros after they were displaced. When the placer attempts to reduce the overlap and converge into a legal position, it requires displacing some standard cells that would naturally be close to the macros into far corners, as seen in Figure 5.8.

Regardless, the full placement results indicate two main things:

1. The global floorplanning stage is the most relevant. Placing the standard cells correctly and moving the macros within the defined region contribute only a small % of improvements over the final HPWL. The main driver in determining if a design will be good or not, is correctly defining the

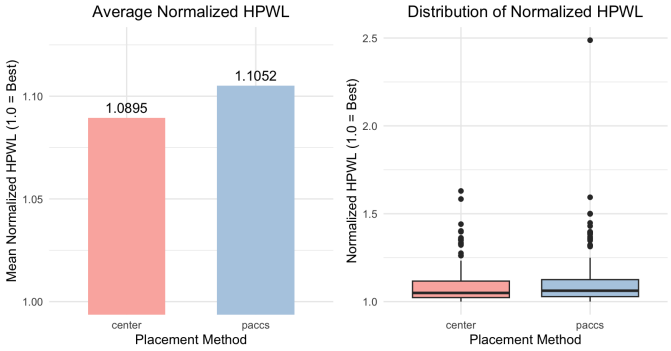


Figure 5.7: Comparison of Normalized HPWL when placing macros at the center of the blocks vs when displaced using PACCS.

horizontal and vertical constraints of each block, and optimizing their physical coordinates.

2. The most important thing when optimizing the design is to have the macros as close as possible to their most affine components. For this reason, two things that could be very relevant were not implemented. First, as mentioned before, finding a better initial placement for standard cells which then serves as a centroid reference point for displacing the macro within the blocks. Second, doing an incremental global placement where the standard cells have an initial estimated position, and pre-defined regions to stay close to the previously optimized partitions.

Finally, looking at the best results found for each benchmark compared to the results of various research and industrial placers (taken from [28]) the first thing to note is the reliability. Even if the floorplanning stage failed for *newblue3*, it proved to be reliable for all the other benchmarks. Comparatively NTUPlace3 and Kraftwerk failed more often.

Furthermore, while some of the state-of-the-art placers like mPL6 or FLOP demonstrate a superior HPWL, the placement framework proposed here proved to be very stable, and only an 11% worse on average than the best placements. The framework used also demonstrated it can be very scalable with larger designs, outperforming every other framework with *newblue5* and being only a 1.5% worse than the best for *newblue6*. The largest gap in the results came from some of the smaller circuits (*adaptec2*, *bigblue3*).

A final important note, however, is that the best performing algorithm (FLOP) also allows rotating macros and takes advantage of incremental placement. When the outputs are compared with the reported results of FLOP-NR and FLOP-NI, our placement framework is only outperformed by a 5%.

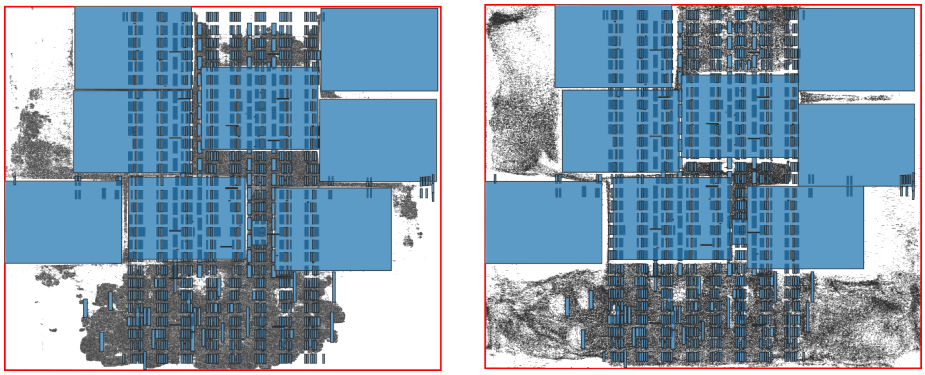


Figure 5.8: Example of a problematic placement of Standard Cells in *newblue2*. To the left the legalized solution when macros are the center of the blocks. To the right the solution when using PACCS, which displaces the cells to the corners likely because of a lack of white space between macros.

Benchmark	**Best HPWL (Center)**	**Best HPWL (PACCS)**	**Normalized (Center)**	**Normalized (PACCS)**
adaptec1	82060379	**82017434**	1.0005	1
adaptec2	**111515573**	112652793	1	1.0102
adaptec3	**197110229**	197242011	1	1.0007
adaptec4	**182040152**	182309114	1	1.0015
adaptec5	418387757	-	1	-
bigblue1	**107262126**	108580789	1	1.0123
bigblue2	147997462	**147833094**	1.0011	1
bigblue3	**429821534**	437344904	1	1.0175
bigblue4	**835335772**	-	1	-
newblue1	**72129214**	75087202	1	1.0410
newblue2	**226938254**	228150023	1	1.0053
newblue4	**307716952**	310174446	1	1.0080
newblue5	**498838900**	499960384	1	1.0022
newblue6	**489984933**	-	1	-
newblue7	**1134754785**	-	1	-
Norm			**1**	**1.0089**

Table 5.4: Comparison of Overall Best HPWL and Normalized Values when placing macros with and without PACCS. At the bottom row, the average normalized performance.

Benchmark	**Best Found**	**NTUplace3** [22]	**mPL6** [47]	**Capo10.5** [48]	**Kraftwerk** [45]	**FLOP** [28]
adaptec1	82.02	80.45	77.84	84.77	86.73	**76.83**
adaptec2	111.52	136.46	88.40	92.61	108.61	**84.14**
adaptec3	197.11	-	180.64	202.37	272.76	**175.99**
adaptec4	182.04	177.23	162.02	202.38	260.96	**161.68**
bigblue1	107.26	95.11	99.36	112.58	130.78	**94.92**
bigblue2	147.83	144.42	**144.37**	149.54	-	153.02
bigblue3	429.82	-	**319.63**	583.37	417.73	346.24
bigblue4	835.34	**764.72**	804.00	915.73	969.03	777.84
adaptec5	418.39	-	376.30	565.88	402.21	**357.83**
newblue1	72.13	**60.73**	66.93	110.54	76.22	67.97
newblue2	226.94	-	**179.18**	303.25	272.67	187.40
newblue3	-	-	415.86	1282.19	374.54	**345.99**
newblue4	307.72	-	277.69	300.69	291.45	**256.54**
newblue5	**498.84**	-	515.49	570.32	503.13	510.83
newblue6	489.98	-	**482.44**	609.16	651.34	493.64
newblue7	1134.75	-	**1038.66**	1481.45	-	1078.18
Norm	**1.11**	**1.08**	**1.02**	**1.44**	**1.26**	**1**

Table 5.5: HPWL Comparison of various placers on MMS Benchmarks ($\times 10^6$).

Chapter 6

Discussion

The results obtained from the proposed mixed-size placement methodology provide significant insights into the delicate relationship between logical netlist partitioning and physical layout optimization. This section evaluates the empirical data against the research hypotheses formulated at the beginning of this study.

6.1 Evaluation of the Placement Framework

Hypothesis $\mathbf{H}_1$: *A hierarchical placement framework that transforms a normalized netlist hypergraph into hybrid macro-standard cell blocks and takes advantage of connectivity information can achieve comparable or lower HPWL compared to other placement flows.*

The experimental results validate $\mathbf{H}_1$. By merging standard cells into hybrid blocks with their most tightly connected macros, the methodology successfully prevented the scattering of high-affinity logic. The normalization of macro weights during the hypergraph partitioning phase ensured that the clustering was made completely by topological connectivity rather than being biased by physical area. Because of this, when the final analytical global placement was executed, standard cells naturally moved towards the macros with the most affinity, resulting in competitive Half-Perimeter Wirelength (HPWL) metrics.

6.2 Hypergraph Partitioning and Metric Tuning

Hypothesis $\mathbf{H}_{2a}$: *The relationship between the target partition parameter k and the final HPWL is non-monotonic; specifically, there is an intermediate configuration ($k > 0$) that isolates clusters of highly connected standard-cell that will outperform a strict macro-anchored-only configuration ($k = 0$).*

As seen from the resutls, the data supports $\mathbf{H}_{2a}$. When partitioning, $k = 0$ achieves the best theoretical logical cuts by attaching every standard cell to a macro. However, this often results in very large, rigid hybrid blocks that restrict

the floorplanner's spatial flexibility. On the other hand, high values of k result in a bigger degradation connectivity of the graph partitions. The results showed that on average, a non-monotonic behavior with an intermediate k configuration can reach an optimal balance. It sacrifices some logical connectivity to provide the global placer with smaller, independent standard-cell clusters, ultimately yielding the best final HPWL.

Hypothesis $\mathbf{H}_{2b}$: *Increasing the allowable area imbalance (ϵ) during the multi-way partitioning phase relaxes topological constraints, allowing the floorplanner more flexibility to satisfy fixed-outline and aspect ratio constraints, which reduces the complexity of final overlap legalization penalties.*

The results confirm $\mathbf{H}_{2b}$, but also present an interesting physical paradox. That is, a lower imbalance would seemingly mean that all blocks have more similar physical areas, resulting in a smaller block weight variance (CV). Nevertheless, the analysis done proved that the disparity in each circuit is guided more by the inherent distribution of sizes in standard cells and the net topology.

Then, relaxing the target imbalance (ϵ) granted the KaHyPar engine the freedom required to find cleaner logical cuts. Therefore, artificially restricting ϵ only forced the partitioner to make highly penalized cuts without actually improving the physical block variance. Relaxing ϵ allowed different-sized blocks to naturally form, giving more freedom to the aspect ratios then used by the floorplanner.

Hypothesis $\mathbf{H}_{2c}$: *Advanced hypergraph partitioning objectives that account for net pins, such as Sum of External Degrees (SOED) and KM-1, will be more accurate predictors of the quality in the final physical layout (HPWL) than the standard Edge-cut metric.*

This hypothesis is supported by the correlation data. While the standard Edge-cut metric correlates with general layout quality, it is a binary measurement that does not take into account the spatial reality of physical design. Because HPWL measures physical bounding boxes, a cut net that spans five different layout blocks will have a much higher routing penalty than a net spanning only two. By explicitly accounting for multi-block spans, the KM-1 and SOED metrics proved to be more related to the final analytical HPWL.

Still, none of the hypergraph metrics had a strong correlation to the quality of the design, and more importantly, it varied a lot from one benchmark into another. Therefore, while it is possible to say that there is a association, none of them would serve as a reliable predictor of a design's quality.

6.3 Mixed-Sized Placement

Hypothesis $\mathbf{H}_{3a}$: *Placing macros at the center of the pre-partitioned hybrid blocks will result in a worse quality placement compared to displacing the macros according to the pull forces of the standard cells and their pin locations.*

The results of this work do not fully support this hypothesis. Even though the PACCS implementation follows intuitive ideas for reducing the wirelength (like moving macros closer to the pins they connect to), the empirical data

suggests that staying as close as possible to the components clustered together is a more effective strategy. This is consistent with the partitioning, and in fact, suggests that the obtained partitions were good at minimizing the objective of external connectivity cuts in the first place. Although more experimentation is needed, and intuitively placing the macros at the center of the blocks shouldn't always be optimal, the implementation proposed in this work failed to take advantage of the clusters.

Another possibility for why displacing the macros did not improve upon placing them at the center is that the floorplanner does not consider the relative positions of the pins when minimizing wirelength. That is, the floorplan legalization and optimization is performed by measuring the HPWL always from the center of the blocks. While this difference shouldn't have a big impact, it is possible that it discourages moving the macros away from said center.

Hypothesis $\mathbf{H}_{3b}$: *Guiding the analytical global placement of standard cells using the pre-defined regions derived from legalized hybrid blocks reduces the continuous optimization search space, resulting in fewer Nesterov gradient descent iterations, faster convergence, and a lower final HPWL compared to a flat, unconstrained global placement flow.*

Finally, although it was initially considered, incremental placement of standard cells was not fully implemented. Partially because of the limitations of the open-source placement tools tested, and partially because of the time constraints in which the project was done. It remains as an open question for future research, as at least other placers (FLOP vs FLOP-NI [28]) have demonstrated to get some benefits from it.

Chapter 7

Conclusions

This thesis proposed, developed, and evaluated a comprehensive, multi-stage mixed-size placement methodology for modern VLSI designs. The general goal was to tie together the logical graph partitioning and the continuous analytical placement. Based on the experimental results and the validation of the research hypotheses, the specific objectives of this work have been successfully fulfilled:

- **Connectivity-Driven Clustering:** A robust strategy was successfully developed to cluster standard cells into hybrid blocks anchored by macros. By applying weight normalization to the macro vertices during the hypergraph construction, the partitioning engine was forced to optimize strictly for logical affinity. This guarantees that highly connected logic stays physically close throughout the layout process, which reduces long wires.
- **Hypergraph Analysis:** The methodology provided a deep empirical evaluation of multi-way partitioning metrics. Regardless, the analysis concludes that even if routing-aware metrics are superior to the traditional Edge-cut metric, none of them are reliable predictors of a design's quality.
- **Constraint-Aware Floorplanning:** Existing multi-stage floorplanning algorithms were successfully adapted to handle the reconstructed hybrid blocks. The flow successfully handled the variances in block sizes under strict fixed-outline and aspect ratio constraints guided by the macros.
- **Region-Aware Standard Cell Placement:** As mentioned in the discussion, this objective was not completely fulfilled. It remains as an open question for future research.
- **Benchmark Validation:** The entire methodology was validated using the industry-standard ISPD 2005 and 2006 mixed-size benchmarks, and compared against other state-of-the-art algorithms and placers. By extracting and analyzing metrics across the changing parameters (k-values,

imbalance targets, and block weight variances), the proposed flow demonstrated its viability for further development, and its capability to achieve competitive layouts.

Ultimately, this research demonstrates that treating mixed-size placement not exclusively as a flat mathematical optimization, but as a physical design flow driven by the netlist topology is a good way to manage the extreme complexity of modern VLSI circuits. By intelligently clustering, partitioning, and spatially guiding the components before applying more traditional analytical algorithms, the proposed methodology successfully produces high-quality layouts.

Chapter 8

Future Work

The mixed-size placement methodology presented in this thesis establishes a good initial framework for handling modern VLSI designs. However, physical design is a highly complex optimization problem, and there are still many ways in which the results could be expanded, refined, and optimized. This chapter discusses some key ideas for future research, focusing on improving logical partitioning, floorplanning, macro-shifting mechanics, and improving the input before the use of analytical global placement.

8.1 Hypergraph Partitioning

In the netlist hypergraph modeling stage, macro vertices are assigned a uniform weight of $w(v_i) = 1$ to prevent the partitioning engine from becoming biased by the huge differences in physical area. While this successfully prioritizes the nets topology, it completely decouples the partitioning process from physical area constraints. This can cause blocks with very big area imbalances that later complicate the floorplanning stage.

To solve this, future work can evaluate implementing a Logarithmic Weight Scaling strategy for macro vertices. Instead of setting them all to 1, macro weights can be scaled relative to the average standard cell area using a logarithmic function:

$$w(v_i) = 1 + \beta \cdot \ln\left(\frac{A_{\text{macro}}(v_i)}{A_{\text{cell,avg}}}\right), \quad \forall v_i \in M \tag{8.1}$$

where $\beta \geq 0$ can be used as a tuning parameter controlling the influence of physical area on the partitioning engine. When $\beta = 0$, the model reduces to the same uniform baseline of every weight being 1. As β increases, the partitioner is forced to distribute less standard cells into blocks that already have huge macros, which can result in more balanced physical dimensions.

Alternatively, future work can also explore using other metrics for the capacity constraints during hypergraph partitioning. Rather than measuring block

balance only through the size area (ϵ), metrics such as pin-density and terminal connection density could be used as the constraints for clustering. This would help with blocks being balanced not just by physical size, but by their routing demand and local congestion limits. If too many connections end up in one block, it could be a cause for routing issues later on. Similarly, if a block has a lot of connections with terminals in the border, this information could help improve the floorplanning process by directly putting it closer to these terminals.

8.2 Floorplanning Configurations

The floorplanning tool utilized in this work possesses two capabilities that were not fully exploited.

First, we are able to define more complex geometric shapes based on their rectangle configurations. Specifically, instead of modeling each block as a single rigid rectangle, blocks can be configured as a multi-rectangle structure that consists of a core central rectangle and up to four peripheral rectangles situated on the borders (top, bottom, left, and right), each with independent aspect ratios and sizing limits. If the size of the macro is used as the central rectangle, the area added by the standard cells can then be distributed in a way that frees up the blocks' corners (Figure 8.1).

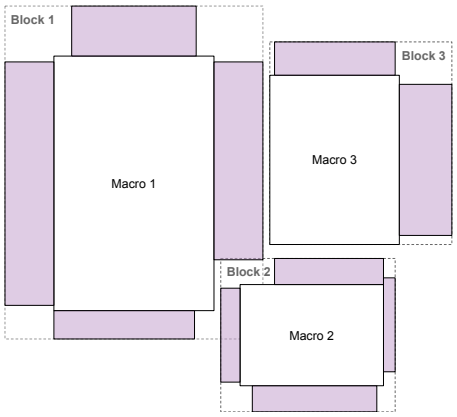


Figure 8.1: Sample diagram showing how blocks defined through multiple rectangles could produce a legal placement, that would be illegal otherwise.

Second, the die configuration can also be setup in different regions defined by rectangles, and some of these regions can be either used as obstacles where nothing can be placed, or used solely for terminals. This would be particularly useful for benchmarks such as *newblue2* and *newblue3* that have very dense clusters of area-array I/O pins in the center. A sample diagram of a complex die configuration, inspired in the original placement of *newblue3*, that could improve the quality of the design is shown in Figure 8.2.

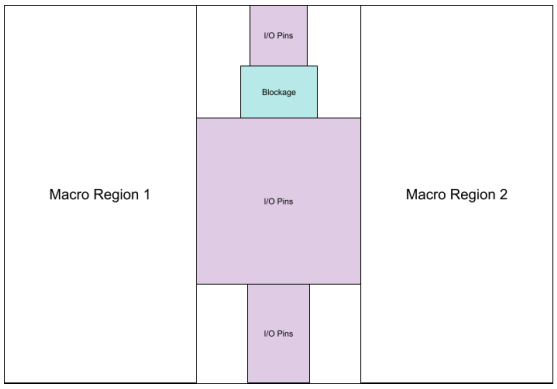


Figure 8.2: Sample diagram showing how regions defined through multiple rectangles could improve the floorplanning configuration of *newblue3*.

Finally, in [28] they mention their placer achieves much higher quality results when they allow rotation of macros. Even though in industry standards this is uncommon since very large nets like V_{cc} and Ground tend to have strict alignments, and this is currently not possible in the floorplanner used because it only constraints blocks into a minimum and maximum aspect ratio; it would be a good branch for research. Many of the more constrained benchmarks could find feasible solutions more easily, and some rotations might greatly improve the overall routing.

8.3 PACCS Refinement

The Pin-Affinity and Cell-Centroid Shifting (PACCS) algorithm shown in the methodology introduces a connectivity driven approach to place macros within the partitioned and legalized blocks. However, as mentioned in the way it was applied, the optimal physical position of a macro is highly dependent on the locations of its local standard cells, but in the flow used here, these cells are not being placed until later. Two strategies to address this can be explored:

1. **External Gravity Point Estimation:** Before shifting the macro, the logical standard cells P_i, can be treated as a single entity. Then, an External Gravity Point, $G_{\text{ext}}(P_i)$, can be computed by evaluating the physical coordinates of all fixed external blocks, macro pins, and I/O terminals connected to P_i:

$$G_{\text{ext}}(P_i) = \frac{\sum_{u \notin B_i} \kappa(P_i, u) \cdot \vec{x}(u)}{\sum_{u \notin B_i} \kappa(P_i, u)} \tag{8.2}$$

   where $\kappa(P_i, u)$ represents the connectivity weight (number of shared nets) between cluster P_i and external object u, and $\vec{x}(u)$ is the absolute coordinate of u. The centroid $C(P_i)$ in the PACCS formulation would then

be anchored to this external gravity vector, pushing the macro away from where the standard cells are pulled.

2. **In-Block Quadratic Pre-Placement:** A localized, unconstrained quadratic placement can be executed solely within the bounding box of block B_i. By modeling the internal nets of the block as a system of classical springs, a system of sparse linear equations can be solved to generate temporary, overlapping coordinates for the cells in P_i. Even if the positions are illegal, these coordinates can provide a good initial estimate for the centroid $C(P_i)$.

8.4 Standard Cells Incremental Placement

In the final stage of the methodology, standard cells are released from their blocks, and global placement is executed using the Nesterov-based analytical solver in RePlAce. Commonly, analytical placers initialize all standard cells at the bottom-left corner of the die $(0, 0)$, and it was also the case in this study.

An idea to improve these placements and the convergence to a more optimal solution in less time would be to use a localized spatial initialization method. Standard cells could start within a loosely bounded region that corresponds with the physical coordinates of the finalized blocks B_i to which they were logically assigned during partitioning.

By starting the analytical placement with standard cells already roughly dispersed in their regions:

- The global placer should require fewer iterations to resolve density overlap, speeding up convergence times.
- The local logical clusters preserved during floorplanning will be maintained, preventing the Nesterov gradient from pulling closely connected cells across the die, and ideally producing a placement with better total wirelength (HPWL) and fewer routing violations.

Bibliography

[1] "Physical design (electronics)," May 2026, page Version ID: 1353228758. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Physical_design_(electronics)&oldid=1353228758

[2] D. Sylvester and K. Keutzer, "A global wiring paradigm for deep submicron design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 2, pp. 242–252, Feb. 2000. [Online]. Available: https://ieeexplore.ieee.org/document/828553/

[3] M. Khasawneh and P. H. Madden, "What's So Hard About (Mixed-Size) Placement?" in *Proceedings of the 2022 International Symposium on Physical Design*, ser. ISPD '22. New York, NY, USA: Association for Computing Machinery, Apr. 2022, pp. 57–64. [Online]. Available: https://dl.acm.org/doi/10.1145/3505170.3511035

[4] D. F. Wong and C. L. Liu, "A new algorithm for floorplan design," in *Proceedings of the 23rd ACM/IEEE Design Automation Conference*, ser. DAC '86. Las Vegas, Nevada, USA: IEEE Press, Jul. 1986, pp. 101–107. [Online]. Available: https://dl.acm.org/doi/10.5555/318013.318030

[5] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1518–1524, Dec. 1996. [Online]. Available: https://ieeexplore.ieee.org/document/552084

[6] S. Adya and I. Markov, "Fixed-outline floorplanning: enabling hierarchical design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 6, pp. 1120–1135, Dec. 2003. [Online]. Available: https://ieeexplore-ieee-org.recursos.biblioteca.upc.edu/document/1255486

[7] H. Murata and E. S. Kuh, "Sequence-pair based placement method for hard/soft/pre-placed modules," in *Proceedings of the 1998 international symposium on Physical design*. Monterey California USA: ACM, Apr. 1998, pp. 167–172. [Online]. Available: https://dl.acm.org/doi/10.1145/274535.274560

[8] S. Schlag, T. Heuer, L. Gottesbüren, Y. Akhremtsev, C. Schulz, and P. Sanders, "High-Quality Hypergraph Partitioning," *ACM J. Exp. Algorithmics*, vol. 27, pp. 1.9:1–1.9:39, Feb. 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3529090

[9] Z. Donovan, G. Gutin, V. Mkrtchyan, and K. Subramani, "Clustering without replication in combinatorial circuits," *Journal of Combinatorial Optimization*, vol. 38, no. 2, pp. 481–501, Aug. 2019. [Online]. Available: https://doi.org/10.1007/s10878-019-00394-1

[10] E. Lawler, K. Levitt, and J. Turner, "Module Clustering to Minimize Delay in Digital Networks," *IEEE Transactions on Computers*, vol. C-18, no. 1, pp. 47–57, Jan. 1969. [Online]. Available: https://ieeexplore.ieee.org/document/1671117/

[11] G. Karypis and V. Kumar, "A Hypergraph Partitioning Package."

[12] I. Bustany, A. B. Kahng, I. Koutis, B. Pramanik, and Z. Wang, "K-SpecPart: Supervised embedding algorithms and cut overlay for improved hypergraph partitioning," Jun. 2023, arXiv:2305.06167 [cs.LG]. [Online]. Available: http://arxiv.org/abs/2305.06167

[13] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph Neural Networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 3558–3565, Jul. 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4235

[14] X. Fang, C. Huan, B. Wang, S. Ma, H. Zhang, and C. Zhao, "HyperSF: A Hypergraph Representation Learning Method Based on Structural Fusion," in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2025, pp. 1–5, iSSN: 2379-190X. [Online]. Available: https://ieeexplore.ieee.org/document/10887924/

[15] R. Otten, "Automatic Floorplan Design," Jul. 1982, pp. 261–267.

[16] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-trees: a new representation for non-slicing floorplans," in *Proceedings 37th Design Automation Conference*, Jun. 2000, pp. 458–463. [Online]. Available: https://ieeexplore.ieee.org/document/855354

[17] P.-N. Guo, T. Takahashi, C.-K. Cheng, and T. Yoshimura, "Floorplanning using a tree representation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 2, pp. 281–289, Feb. 2001. [Online]. Available: https://ieeexplore.ieee.org/document/908471/

[18] I. L. Markov, J. Hu, and M.-C. Kim, "Progress and challenges in VLSI placement research," in *Proceedings of the International Conference on Computer-Aided Design*, ser. ICCAD '12. New York, NY, USA: Association for Computing Machinery, Nov. 2012, pp. 275–282. [Online]. Available: https://dl.acm.org/doi/10.1145/2429384.2429441

[19] M. A. Breuer, "A class of min-cut placement algorithms," in *Papers on Twenty-five years of electronic design automation*. ACM, Jun. 1988, pp. 142–148. [Online]. Available: https://dl.acm.org/doi/10.1145/62882.62896

[20] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Papers on Twenty-five years of electronic design automation*, ser. 25 years of DAC. New York, NY, USA: Association for Computing Machinery, Jun. 1988, pp. 241–247. [Online]. Available: https://dl.acm.org/doi/10.1145/62882.62910

[21] T.-C. Chen, P.-H. Yuh, Y.-W. Chang, F.-J. Huang, and T.-Y. Liu, "MP-Trees: A Packing-Based Macro Placement Algorithm for Modern Mixed-Size Designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 9, pp. 1621–1634, Sep. 2008. [Online]. Available: https://ieeexplore.ieee.org/document/4603071

[22] T.-C. Chen, T.-C. Hsu, Z.-W. Jiang, and Y.-W. Chang, "NTUplace: a ratio partitioning based placement algorithm for large-scale mixed-size designs," in *Proceedings of the 2005 international symposium on Physical design*, ser. ISPD '05. New York, NY, USA: Association for Computing Machinery, Apr. 2005, pp. 236–238. [Online]. Available: https://dl.acm.org/doi/10.1145/1055137.1055188

[23] J. Lu, H. Zhuang, P. Chen, H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng, and C.-K. Cheng, "ePlace-MS: Electrostatics-Based Placement for Mixed-Size Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 5, pp. 685–698, May 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7008518

[24] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang, "RePlAce: Advancing Solution Quality and Routability Validation in Global Placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 9, pp. 1717–1730, Sep. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8418790

[25] "The-OpenROAD-Project/OpenROAD," Jun. 2026, original-date: 2019-10-28T17:48:14Z. [Online]. Available: https://github.com/The-OpenROAD-Project/OpenROAD

[26] A. Vidal-Obiols, J. Cortadella, J. Petit, M. Galceran-Oms, and F. Martorell, "RTL-Aware Dataflow-Driven Macro Placement," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2019, pp. 186–191, iSSN: 1558-1101. [Online]. Available: https://ieeexplore.ieee.org/document/8714812/

[27] ——, "Multilevel Dataflow-Driven Macro Placement Guided by RTL Structure and Analytical Methods," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 12, pp.

2542–2555, Dec. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9309318/

[28] J. Z. Yan, N. Viswanathan, and C. Chu, "Handling complexities in modern large-scale mixed-size placement," in *2009 46th ACM/IEEE Design Automation Conference*, Jul. 2009, pp. 436–441, iSSN: 0738-100X. [Online]. Available: https://ieeexplore.ieee.org/document/5227031/

[29] Gi-Joon Nam, S. Reda, C. Alpert, P. Villarrubia, and A. Kahng, "A fast hierarchical quadratic placement algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 4, pp. 678–691, Apr. 2006. [Online]. Available: http://ieeexplore.ieee.org/document/1610733/

[30] X. Zhao, Z. Li, Y. Cai, J. Chen, Y. Pan, and X. Guo, "DARE: Enriching Physical Dataflow Awareness for Macro Placement Optimization," *ACM Trans. Des. Autom. Electron. Syst.*, Jan. 2026, just Accepted. [Online]. Available: https://dl.acm.org/doi/10.1145/3795530

[31] J.-M. Lin, Y.-L. Deng, Y.-C. Yang, J.-J. Chen, and P.-C. Lu, "Dataflow-Aware Macro Placement Based on Simulated Evolution Algorithm for Mixed-Size Designs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 5, pp. 973–984, May 2021. [Online]. Available: https://ieeexplore-ieee-org.recursos.biblioteca.upc.edu/document/9360844

[32] C.-K. Cheng, A. B. Kahng, S. Kundu, Y. Wang, and Z. Wang, "An Updated Assessment of Reinforcement Learning for Macro Placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/11300304

[33] Y. Wang, H. Gu, X. Zheng, K. Peng, and Z. Zhu, "Late Breaking Results: RL-Based Macro Placement with Cell Clustering and Rudy Modeling for Routability Optimization," 2026.

[34] Y. Du, R. Yan, F. Qu, and J. Tan, "CHIP-MAP: A Collaborative Optimization Framework for Macro Placement Using Large Language Models," 2026.

[35] Y. Lin, S. Dhar, W. Li, H. Ren, B. Khailany, and D. Z. Pan, "DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement."

[36] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz, "The ISPD2005 placement contest and benchmark suite," in *Proceedings of the 2005 international symposium on Physical design*, ser. ISPD '05. New York, NY, USA: Association for Computing Machinery, Apr. 2005, pp. 216–220. [Online]. Available: https://dl.acm.org/doi/10.1145/1055137.1055182

[37] "The ISPD98 Circuit Benchmark Suite." [Online]. Available: https://vlsicad.ucsd.edu/UCLAWeb/cheese/ispd98.html#Benchmark%20File%20Format

[38] "LEF/DEF Language Reference." [Online]. Available: https://www.ispd.cc/contests/18/lefdefref.pdf

[39] A. B. Kahng, M. Kim, S. Kim, and M. Woo, "RosettaStone: Connecting the Past, Present, and Future of Physical Design Research," *IEEE Design & Test*, vol. 39, no. 5, pp. 70–78, Oct. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9785815/

[40] S. Kumar, L. Pires, S. Ponnuswamy, C. Nanavati, J. Golusky, M. Vojta, S. Wadi, D. Pandalai, and H. Spaanenberg, "A benchmark suite for evaluating configurable computing systems—status, reflections, and future directions," in *Proceedings of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate arrays*, ser. FPGA '00. New York, NY, USA: Association for Computing Machinery, Feb. 2000, pp. 126–134. [Online]. Available: https://dl.acm.org/doi/10.1145/329166.329193

[41] A. B. Kahng, H. Lee, and J. Li, "Horizontal benchmark extension for improved assessment of physical CAD research," in *Proceedings of the 24th edition of the great lakes symposium on VLSI*. Houston Texas USA: ACM, May 2014, pp. 27–32. [Online]. Available: https://dl.acm.org/doi/10.1145/2591513.2591540

[42] C. J. Alpert and View Profile, "The ISPD98 circuit benchmark suite," in *Proceedings of the 1998 international symposium on Physical design*, ser. ACM Conferences, Apr. 1998, pp. 80–85. [Online]. Available: https://dl.acm.org/doi/10.1145/274535.274546

[43] Y. Yimamu, G. Pastor-Rue, and J. Cortadella, "Chip Floorplanning Combining Convex and Non-convex Optimization," Mar. 2026.

[44] J. Cortadella, "jordicf/CPUPC," Jun. 2026, original-date: 2025-11-28T07:45:27Z. [Online]. Available: https://github.com/jordicf/CPUPC

[45] B. Obermeier, View Profile, H. Ranke, View Profile, F. M. Johannes, and View Profile, "Kraftwerk," in *Proceedings of the 2005 international symposium on Physical design*, ser. ACM Conferences, Apr. 2005, pp. 242–244. [Online]. Available: https://dl.acm.org/doi/10.1145/1055137.1055190

[46] H. Eisenmann and F. M. Johannes, "Generic Global Placement and Floorplanning."

[47] T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze, and M. Xie, "mPL6: enhanced multilevel mixed-size placement," in *Proceedings of the 2006 international symposium on Physical design*, ser. ISPD '06. New York, NY, USA: Association for Computing Machinery, Apr. 2006, pp. 212–214. [Online]. Available: https://dl.acm.org/doi/10.1145/1123008.1123055

[48] J. Roy, D. Papa, and I. Markov, "Min-cut floorplacement," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 25, pp. 1313–1326, Aug. 2006.

Appendix A

Benchmarks' Original Placements

This section contains the original placement of all macros and I/O terminals from the utilized benchmarks. It does not contain the placement of standard cells.

In the placement for benchmark *newblue1*, all the big macros are freed, and the only fixed components are the I/O pins in the border. As such, there is no "original" placement. The one shown here, corresponds to the one shown in [36].

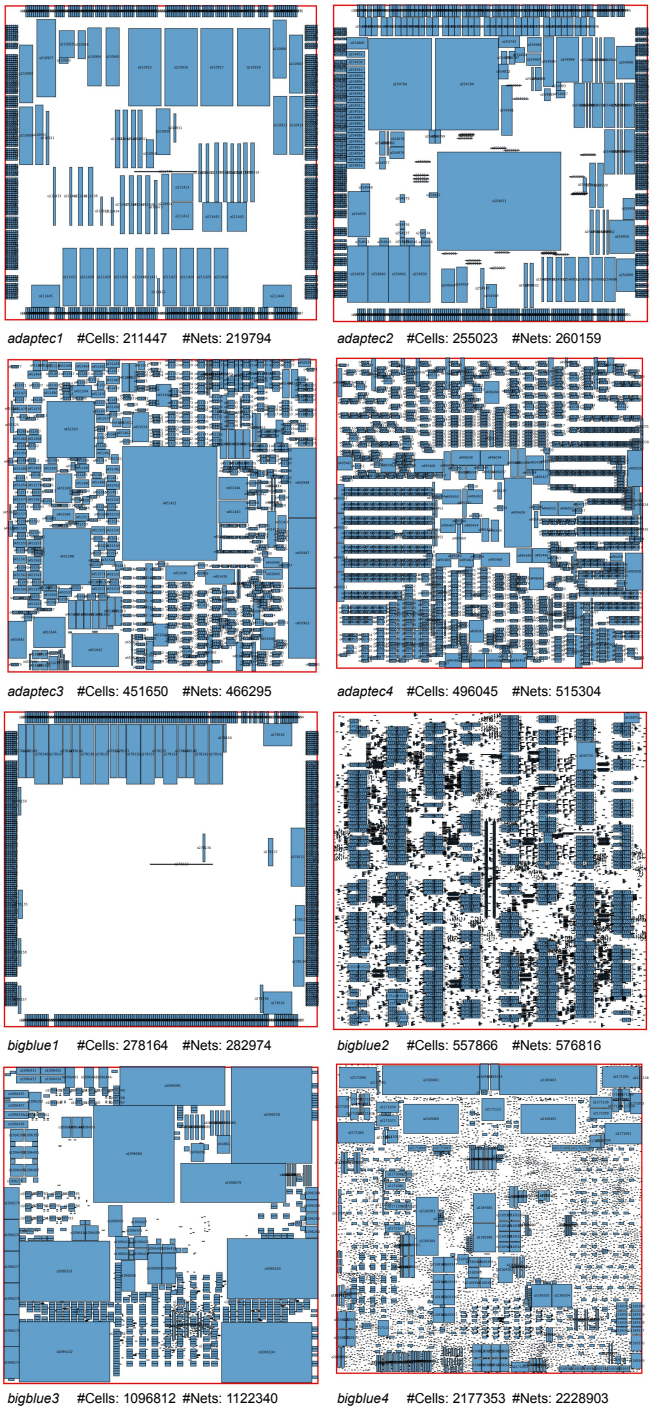


Figure A.1: Original Placement for ISPD 2005 Benchmarks.

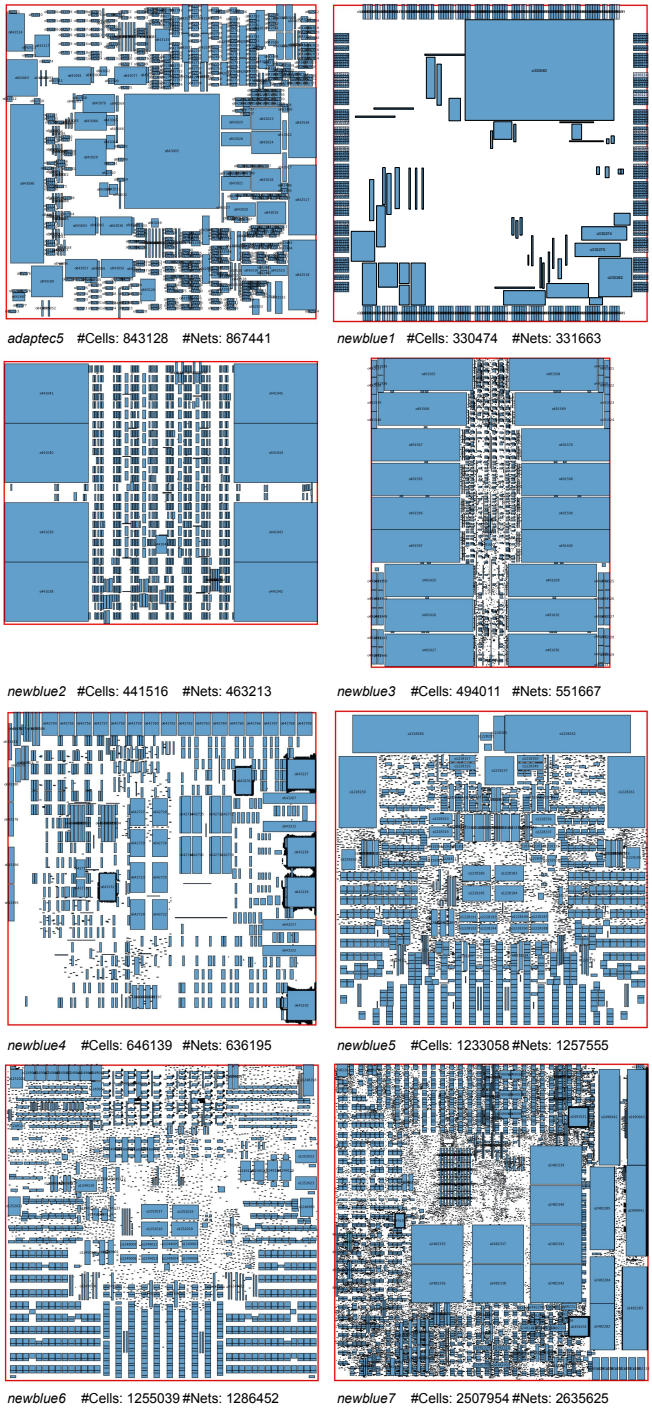


Figure A.2: Original Placement for ISPD 2006 Benchmarks.

Appendix B

Full measured HPWL

First table presents the measured HPWL across all the tested parameters when the macros were placed at the center of the blocks. Second table presents the same measured HPWL when the macros were placed using the PACCS implementation. For each benchmark, the best solution is shown in bold text.

PACCS placement was not run on the biggest benchmarks due to the time limits of this research, and that there did not seem to be any benefit in HPWL.

Benchmark	**Imbalance**	**HPWL by k Value**			
		k = 0	**k = 5**	**k = 10**	**k = 20**
adaptec1	0.3	85089442	82737688	87999424	89623925
	0.4	89895933	85254634	85657911	90287511
	0.5	87902392	83219317	91665019	83813769
	0.6	83615770	84940694	86615024	89234981
	0.7	**82060379**	85756016	82097682	86244895
adaptec2	0.3	114881307	115907068	113436767	117958882
	0.4	-	115901714	121279455	117793300
	0.5	133005377	114965569	120742695	130240610
	0.6	111967964	-	125623084	117377997
	0.7	116569560	113937876	**111515573**	128620280
adaptec3	0.3	212537882	**197110229**	207981527	199393658
	0.4	200008813	200947800	224890935	207547823
	0.5	204869530	204280567	208295264	201148681
	0.6	210263053	221916987	202099318	203357137
	0.7	213475621	211675424	211786960	212918872
adaptec4	0.3	190256148	186708911	185078114	193988166
	0.4	193454693	183262787	188679337	187408685
	0.5	185615695	186946938	186858451	194263663
	0.6	192253670	185995550	185026319	194846582
	0.7	186539148	**182040152**	184465735	182499046
adaptec5	0.3	463697624	432429833	448346354	446120161
	0.4	484703624	442489077	429833575	471264458
	0.5	439143502	418866087	464928768	437958187
	0.6	443406158	**418387757**	427720575	440846536
	0.7	485878951	420009281	448804061	448934637
bigblue1	0.3	115176159	111894313	109078120	116040071
	0.4	114598540	108282781	109585802	115183921
	0.5	111618001	112648325	108407631	115956045
	0.6	112329012	110839253	108748064	111516112

Continued on next page

Table B.1 – continued from previous page

Benchmark	Imbalance	HPWL by k Value			
		k = 0	**k = 5**	**k = 10**	**k = 20**
	0.7	110735907	**107262126**	109691336	111927131
bigblue2	0.3	148466077	149794616	148252530	150029098
	0.4	148673495	151671670	150979208	-
	0.5	148808342	151334144	149209435	-
	0.6	**147997462**	150734047	150125729	148284347
	0.7	148585814	-	150668951	149021890
bigblue3	0.3	509653431	480547763	448314704	680739822
	0.4	461345085	524207746	501429193	-
	0.5	499089575	**429821534**	568268425	602706763
	0.6	491908694	525004243	542312398	512473179
	0.7	-	700462337	510295021	600757561
bigblue4	0.3	844897871	866822097	915045491	891462032
	0.4	870093296	854250298	893025287	890955529
	0.5	864103848	898123453	**835335772**	862232348
	0.6	893860041	861591229	843032712	851031985
	0.7	856976666	874141169	866913770	915061701
newblue1	0.3	77761286	79959343	74984265	73812047
	0.4	74162757	75497746	-	74360964
	0.5	-	74665586	74230515	84802325
	0.6	86851647	79377228	77417205	75255954
	0.7	82789904	73883103	**72129214**	80538077
newblue2	0.3	327003302	307309877	269382793	302177857
	0.4	306521057	239766226	308767741	302677985
	0.5	306021094	309290077	306739401	286358138
	0.6	-	300995263	258560490	**226938254**
	0.7	-	289487511	279857187	317368976
newblue3	0.3	-	-	-	-
	0.4	-	-	-	-
	0.5	-	-	-	-
	0.6	-	-	-	-
	0.7	-	-	-	-
newblue4	0.3	322694513	316576300	316924058	316292842
	0.4	334881601	333305041	313476131	327296247
	0.5	316107154	317622811	323260768	314815527
	0.6	324166226	317335233	315424112	320666319
	0.7	317385428	311741132	316054721	**307716952**
newblue5	0.3	568423268	589974695	-	591586583
	0.4	560387560	577255701	553764456	565290361
	0.5	578626768	564302918	562664295	535190847
	0.6	562460340	538933461	538191131	538847531
	0.7	536381221	554447880	561143479	**498838900**
newblue6	0.3	503108178	504983605	513950019	503146752
	0.4	493674886	495970285	520927527	496081424
	0.5	495469208	490766564	511309009	498206981
	0.6	503374515	**489984933**	524360643	503043891
	0.7	492693764	495795202	505357183	501342634

Continued on next page

Table B.1 – continued from previous page

Benchmark	Imbalance	HPWL by k Value			
		k = 0	**k = 5**	**k = 10**	**k = 20**
newblue7	0.3	1166406309	1197344425	1143440241	1158034580
	0.4	1202979104	**1134754785**	1189471166	1216227764
	0.5	1179502292	1191757461	1195708773	1200485160
	0.6	1211644848	1196317800	1172731908	1192352476
	0.7	1202002627	1249804894	1213964090	1177542010

Table B.1: Final HPWL Results for Macros placed in the center of the block by Benchmark, Imbalance, and k. Smallest HPWL for each benchmark is in bold.

Benchmark	Imbalance	HPWL by k Value			
		k = 0	**k = 5**	**k = 10**	**k = 20**
adaptec1	0.3	85664676	83000571	88953297	90000656
	0.4	90609205	84826441	85942414	89688217
	0.5	89159553	83227757	91947670	84951532
	0.6	84186696	84708601	87468544	91626160
	0.7	**82017434**	85309499	82251417	87097712
adaptec2	0.3	115636195	114579691	115154431	118461394
	0.4	-	116522926	122764573	119787838
	0.5	132721025	115430022	121550844	131677328
	0.6	112787166	-	127952162	117858913
	0.7	118816037	117168722	**112652793**	128855205
adaptec3	0.3	211673151	**197242011**	211852228	199351618
	0.4	200171060	200913756	223688605	207664971
	0.5	203874728	204803823	207779766	202314845
	0.6	214448928	221738598	202765097	203990903
	0.7	215583108	211947246	211192142	214182831
adaptec4	0.3	190059191	186596287	185286502	193611336
	0.4	193746191	182806988	188794260	187243730
	0.5	186346192	187237496	186385430	195210994
	0.6	191914272	185898407	186033257	195376518
	0.7	186388758	**182309114**	185076918	182396872
bigblue1	0.3	117092987	113833432	110349207	117388267
	0.4	116233462	110451845	110858904	115307150
	0.5	112392537	113613332	109931224	117055405
	0.6	114082285	111722094	109410836	113051431
	0.7	111620022	**108580789**	110644174	112792795
bigblue2	0.3	148499721	149598709	148352500	150217407
	0.4	148739404	151203574	149605706	-
	0.5	148812147	151138038	149139031	-
	0.6	147833094	150856002	150052712	**148127201**
	0.7	148385353	-	150401187	148918484

Continued on next page

Table B.2 – continued from previous page

Benchmark	Imbalance	HPWL by k Value			
		k = 0	**k = 5**	**k = 10**	**k = 20**
bigblue3	0.3	514201431	498384902	450475416	585289148
	0.4	468640304	530292473	527347327	-
	0.5	510388060	**437344904**	564536187	644290695
	0.6	505415236	521405566	514941740	522186962
	0.7	-	685044357	514943921	599541542
newblue1	0.3	84018778	85333491	81215067	78523259
	0.4	76113915	80785175	-	79032369
	0.5	-	78247294	75101016	86327517
	0.6	89361162	81536984	78909429	80185644
	0.7	80855749	83513523	76192116	**75087202**
newblue2	0.3	340645237	307477843	283618097	300098169
	0.4	314974420	246140774	328568364	313793848
	0.5	317199944	564527373	316321222	310849020
	0.6	-	299452462	260378392	**228150023**
	0.7	-	305767925	283442902	324559938
newblue4	0.3	321016548	317935997	317050691	315417850
	0.4	334137188	335183658	316520926	327891478
	0.5	316686703	320092573	324331438	314814723
	0.6	324785849	318197695	316480128	319798700
	0.7	317258301	313059905	317479327	**310174446**
newblue5	0.3	568049168	592682835	-	594381310
	0.4	560594593	579533982	556114031	566438891
	0.5	585250927	563758078	567759927	535338422
	0.6	560577016	542145251	538913367	542794556
	0.7	535588032	558805792	561733510	**499960384**

Table B.2: Final HPWL Results for PACCS placement by Benchmark, Imbalance, and k. Smallest HPWL for each benchmark is bolded.

Appendix C

Best Found Placement for every Benchmark

This section contains the best final placement found of every tested benchmark. The only exception is for *newblue3*, where no feasible solution was found. In light blue all the macros and I/O pins in their fixed final position. In dark blue all the standard cells. The overlapping components seen in these figures correspond to the area-array I/O pins for which the original fixed position was kept, but the size was set to zero while the floorplanning and placement procedures ran.

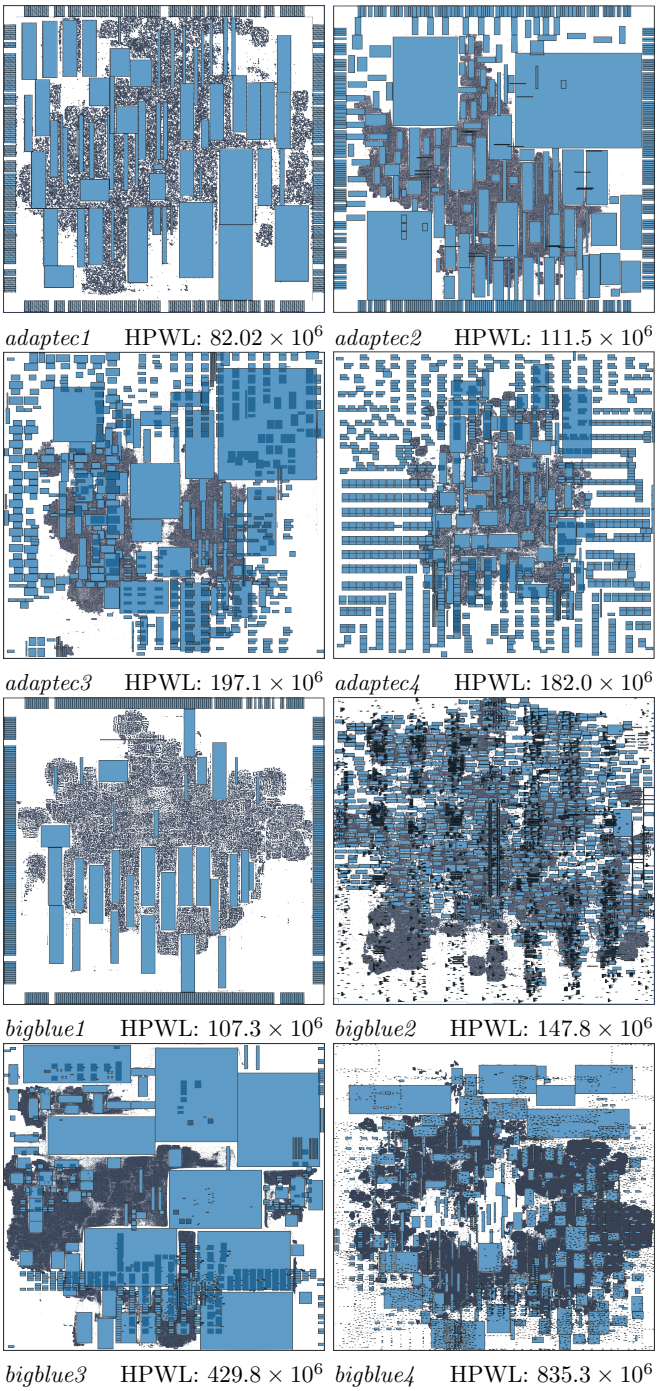


Figure C.1: Best Found Placement for ISPD 2005 Benchmarks.

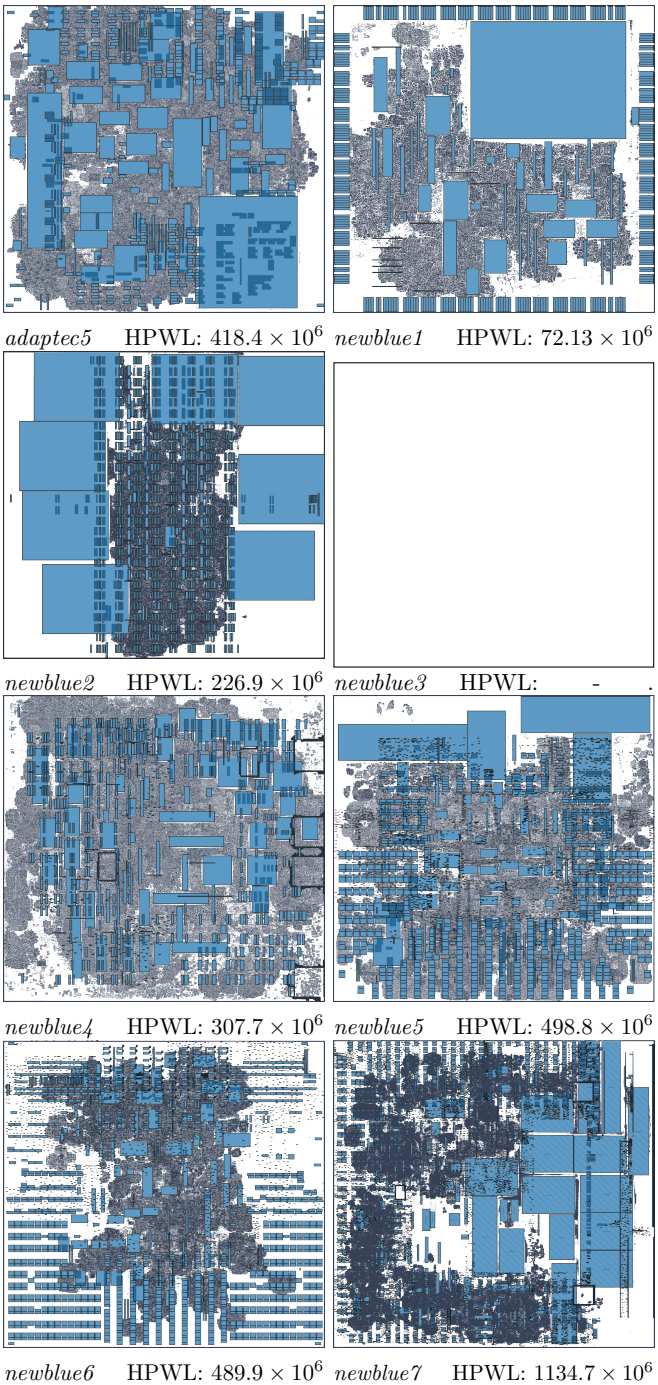


Figure C.2: Best Found Placement for ISPD 2006 Benchmarks.