Floorplanning Optimization for Three-Dimensional Integrated Circuits

Dima Al Saleh

Department of Electrical & Computer Engineering

McGill University

Montréal, Québec, Canada

September, 2024

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Master of Sciences in Electrical Engineering

©2024 Dima Al Saleh

Abstract

Three-dimensional integrated circuits represent a transformative approach to address the challenges of interconnects in modern electronics. By stacking multiple heterogeneous dies and utilizing through-silicon vias, 3D ICs enable efficient vertical connections, reducing the need for long two-dimensional interconnects. Reducing the length of interconnects also reduces the power consumed during communication. This innovative structure not only improves performance but also facilitates the integration of diverse functions on different layers, offering more heterogeneity in the system and becoming a promising avenue for advanced electronic systems.

However, the unique design challenges of 3D ICs have hindered their widespread adoption. Current electronic design automation tools, initially developed for 2D ICs, struggle to address the intricacies of 3D ICs. One critical challenge is floorplanning. Floorplanning is a crucial step in the design process of an IC affecting the area, wirelength, power, delay, and temperature of the chip. During this step, the location and coordinates of each macro to be included in the circuit are determined. Floorplanning becomes even

more crucial for 3D ICs due to the exacerbated thermal problem. The poor thermal conductivity of silicon dioxide in 3D ICs leads to heat buildup and a significant increase in temperature. The power density within the 3D structure is beyond the cooling capacity of conventional air-cooled heat sinks. This has negative consequences for performance and may lead to technical failure.

Moreover, floorplanning in 3D considerably increases the solution space, and the introduction of new optimization criteria, such as the number of through-substrate vias, creates conflicting design objectives.

In this work, a novel P* admissible thermally aware floorplanning algorithm for 3D ICs is presented. The data structures used to represent the floorplans during all stages of the optimization process are matrices. The Matrix Floorplanner is based on manipulations within matrices which support a polynomial optimization and packing time. The algorithm is based on the relative horizontal and vertical relations among the blocks of each layer within the 3D structure. Moreover, the algorithm employs a novel thermal-grid-based thermal cost computation approach. This methodology for temperature computation enables polynomial computation, evaluation, and optimization of solutions. The proposed algorithm meets all criteria to satisfy the P* admissibility. In addition, this work distinguished itself from other P* admissible floorplanners by allowing more types of perturbations, which promotes better diversification during the simulated annealing process. It also simultaneously optimizes area, wirelength, temperature, and the number of TSVs.

The Matrix Floorplanner has been realized in C++ and evaluated on standard floorplanning benchmarks (MCNC and GSRC). Results are compared to previous work and exhibit a significant improvement across performance metrics with respect to similar objective algorithms. For example, for the n100 benchmark, a reduction in area, temperature, and runtime of, respectively, 10.1%, 17.7%, and 89.6% is observed, as compared to previous work.

Furthermore, several netlists from industrial partners were evaluated and exhibited excellent physical and runtime results. With respect to the smallest possible area (no white spaces at all), the matrix floorplanner produces high-quality results with as low as 3.8% white spaces.

The Matrix Floorplanner also outperforms other algorithms, that do not target thermal and TSV optimization, in performance metrics for large floorplans.

Abrégé

Les circuits intégrés (CI) tridimensionnels (3D) représentent une approche transformatrice pour relever les défis des interconnexions dans l'électronique moderne. En empilant plusieurs matrices hétérogènes et en utilisant des vias traversant le substrat (VTS), les CI 3D permettent des connexions verticales efficaces, réduisant ainsi la consommation d'énergie et améliorant les performances. Cette structure innovante facilite également l'intégration de diverses fonctions sur différentes couches, offrant plus d'hétérogénéité dans le système. Cependant, les défis de conception des CI 3D ont entravé leur adoption généralisée. Les outils d'automatisation actuels, initialement développés pour les CI 2D, ont du mal à gérer la planification des étages, une étape cruciale qui affecte la surface, la longueur du fil, la puissance et la température de la puce. Au cours de cette étape, les coordonnées de chaque macro à inclure dans le circuit sont déterminés. La planification des étages devient encore plus cruciale dans les CI 3D en raison du problème thermique exacerbé. La faible conductivité thermique dans les CI 3D entraîne une accumulation de chaleur, ce qui peut affecter les performances et conduire à des défaillances. De plus, la

planification des étages en 3D augmente considérablement l'espace de solution, et l'introduction de nouveaux critères d'optimisation, tels que le nombre de VTS, crée des objectifs de conception contradictoires.

Dans ce travail, un nouvel algorithme de planification des étages, P* sensible à la température pour les CI 3D est présenté. Les structures de données utilisées pour représenter les plans d'étage pendant toutes les étapes du processus d'optimisation sont des matrices. Le Matrix Floorplanner est basé sur des manipulations au sein de matrices qui prennent en charge une optimisation polynomiale et un temps de conditionnement. L'algorithme est basé sur les relations horizontales et verticales relatives entre les blocs de chaque couche au sein de la structure 3D. De plus, l'algorithme utilise une nouvelle approche de calcul des coûts thermiques basée sur une grille thermique. Cette méthodologie de calcul de la température permet le calcul polynomial, l'évaluation et l'optimisation des solutions. L'algorithme proposé répond à tous les critères pour satisfaire à l'admissibilité P*. De plus, ce travail s'est distingué des autres planificateurs d'étages admissibles P* en autorisant davantage de types de perturbations, ce qui favorise une meilleure diversification lors du processus de recuit simulé. Il optimise également simultanément la surface, la longueur des fils, la température et le nombre de VTS.

Le Matrix Floorplanner a été réalisé en C++ et évalué sur des tests de performance de planification d'étage standard (MCNC et GSRC). Les résultats sont comparés aux travaux précédents et montrent une amélioration des mesures de performance par rapport à des

algorithmes objectifs similaires. Les résultats montrent une amélioration significative des mesures de performance clés. Par exemple, pour le test n100, une réduction de la surface, de la température et du temps d'exécution de respectivement 10,1%, 17,7% et 89,6% est observée par rapport aux travaux précédents. De plus, plusieurs listes de connexions de partenaires industriels ont été évaluées et ont montré d'excellents résultats physiques et de temps d'exécution. En ce qui concerne la plus petite surface possible (aucun espace blanc du tout), le Matrix Floorplanner produit des résultats de haute qualité avec seulement 3,8 % d'espaces blancs. Le Matrix Floorplanner surpasse également d'autres algorithmes, qui ne ciblent pas l'optimisation thermique et VTS, dans les mesures de performance pour les grands plans d'étage.

Acknowledgements

Studying at McGill University has been an incredible journey, for which I am deeply grateful. None of this would have been possible without the unwavering support I received.

I may never find the words to fully express my sincere and profound gratitude to Professor Boris Vaisband. He has been a beacon of hope and reassurance whenever I doubted myself or my abilities. Always available to guide me through this tumultuous journey, his knowledge is inspirational and empowering. He diligently and patiently guided the way of my research. Working with him has been an honor that I will carry with me throughout my career.

I feel an enormous amount of gratitude for all the connections and bonds I have formed with my lab friends. Going into this program, I expected to gain much, but I certainly did not anticipate finding a home away from home. Meeting and getting to work with Yousef Safri, Rezvan Mohammad Rezaee, Ataollah Saeed Monir, Scott Fulton, Fahad Rahman Amik, Vincent Zhang, and Okyanus Gumus has been a privilege.

I am deeply grateful to my friends Miriam Boutros, Omar Itani, Nour Soudki, and Tamara Rahmoun, whose inspiring and unique journeys have enriched my own. Though our lives

have taken different directions, their unwavering support and friendship have been a constant source of strength throughout this experience.

I also would like to take this opportunity to thank my parents Rola Doueik and Bassam Al Saleh. Even oceans away, their belief in me, encouragement, and love would resonate with me making any challenge seem trivial and any mountain a mere hill. I also owe to them my love for academia and research as they have always instilled the importance of curiosity and asking questions. I would like to thank my sister, Tala Al Saleh, who somehow always had the right words during the lows and the best cheers during the highs.

Author Contribution

The research presented in this thesis is based on a previously published manuscript [1], in which I was the first author and was solely responsible for the majority of the work. Specifically, I conceptualized the data structure and designed the methodology, including the initialization, P* admissibility, perturbations, and packing. I also designed the evaluation of all metrics during the simulated annealing stage, in particular the fast thermal estimator. Moreover, I implemented the entirety of the floorplanning algorithm in C++, analyzed the data, and wrote the manuscript

The contributions from co-authors included integrating a thermal visualizer ARTSim [2] into the floorplanner to illustrate the obtained solutions, as well as running simulations to expedite the collection of results.

I confirm that the other co-authors have granted me permission to include this work in my thesis and will not use this same manuscript in their respective thesis.

Contents

List of Figures

List of Tables

List of Acronyms

3D	three-dimensional.
AC	ant colony.
AI	artificial intelligence.
AR	aspect ratio.
BTBT	band-to-band tunneling.
CBA	combined bucket and 2D array.
CBA-T	combined bucket and 2D array thermal.
CTE	coefficient of thermal expansion.
GA	genetic algorithms.
GAN	generative adversarial network.
GIDL	gate-included drain leakage.
GNN	graph neural network.
HPWL	half-perimeter wirelength.
IC	integrated circuit.

KOZ	keep-out zone.
ML	machine learning.
NN	neural network.
NP	nondeterministic polynomial.
PSO	particle swarm optimization.
RL	reinforcement learning.
SA	simulated annealing.
SiO_2	silicon dioxide.
SL	supervised learning.
TCG	transitive-closure-graph.
TSV	through-substrate via.
TTSV	thermal through-substrate via.
UL	unsupervised learning.

Chapter 1

Introduction

Further down-scaling devices, to meet the ever-growing computing requirements, presents several electrical and physical challenges that have become increasingly difficult to overcome. Improvements in fabrication processes and the use of different materials, such as high-K dielectric for better channel control, have reached their practical limit. Scaling devices beyond a few tens of nanometers increases significantly parasitic gate currents, gate-induced drain leakage (GIDL), band-to-band tunneling (BTBT), and drain current degradation. Additionally, as devices are scaled down, carrier mobility is diminished, threshold voltage shifts, and trans-conductance decreases [3], [4].

In particular, the increase in leakage current impedes Dennard scaling [5]. The reduction of power density can no longer be achieved, effectively hitting the power wall, thereby causing a plethora of reliability concerns for circuit devices [6]. The conventional practices of voltage

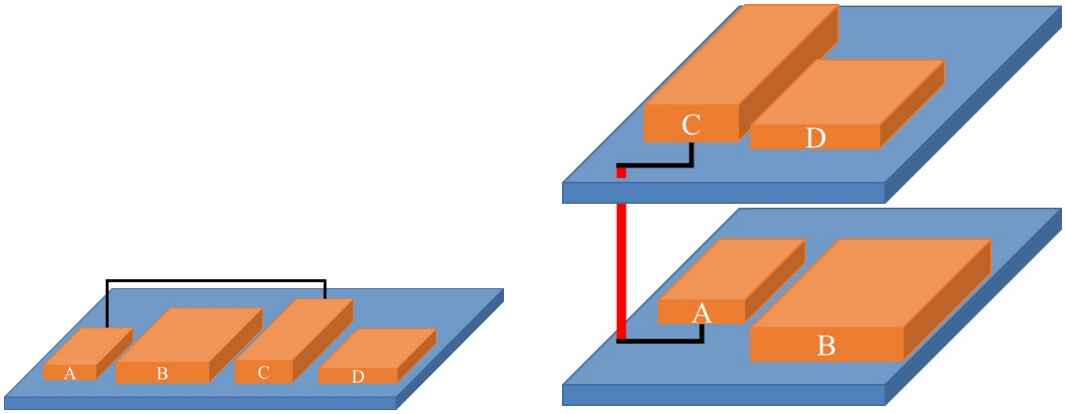
scaling to increase transistor density are, therefore, no longer sufficient.

Amidst these challenges, a paradigm shift is evident, redirecting focus from device scaling towards interconnect scaling. Packaging features have experienced only a 4x scale-down since 1970, in stark contrast to silicon features, which have achieved a 1000x scale reduction (from 10 um to 10 nm since 1970) [7]. This underscores the significant, relatively untapped, potential for enhancements in package scaling. Three-dimensional (3D) integrated circuits (IC) have become a solution with promising potential for this bottleneck. Leveraging advancements in fabrication processes, 3D ICs are more feasible; several companies have already begun commercializing them [8]. However, designing 3D ICs remains a challenging task.

1.1 Benefits of Three Dimensional Circuits

3D ICs are formed by integrating heterogeneous die or wafers on top of one another, creating multiple layers of active devices. These layers are generally connected using through-silicon vias (TSVs) [9]. While monolithic 3D ICs and contactless 3D ICs do not use TSV-based connections, this work will specifically center on TSV-based 3D ICs, as they remain the most popular, feasible, and heterogeneous type of 3D ICs [10]. As depicted in Figure 1.1, integrating blocks vertically increases the density of modules and reduces the need for lengthy interconnects used in conventional two-dimensional platforms.

Decreasing the wirelength of circuit connections has a direct impact on power



(a) **(b)**

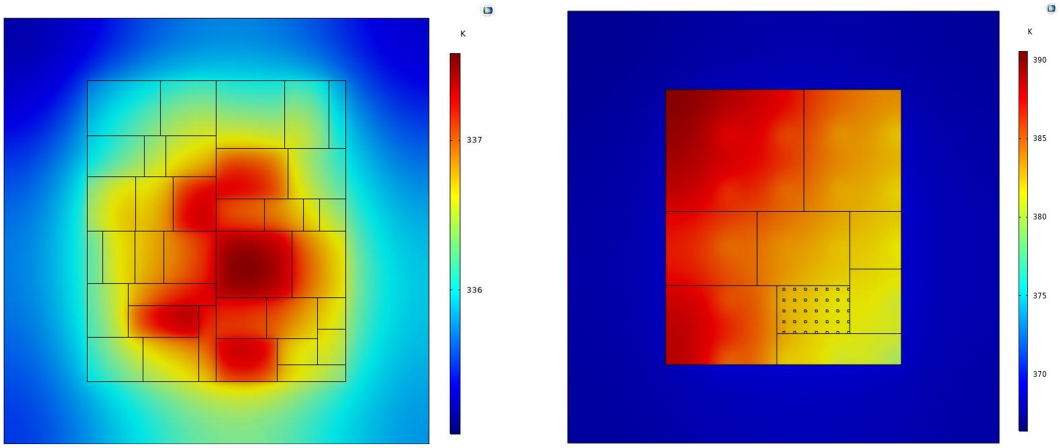
Figure 1.1: A comparative view of a 2D IC and a 3D IC in terms of area and wirelength.

consumption and various other performance metrics of the 3D IC. Shorter interconnects require less energy for signal propagation, resulting in a reduction in interconnect power dissipation [11]. Key performance parameters, such as delay, power, and noise, are predominantly dominated by interconnects rather than device technology [12], [13]. 3D ICs effectively mitigate the interconnect challenge and alleviate interconnect-related performance degradation. Likewise, TSV-based 3D ICs improve the heterogeneity of systems by facilitating the integration of diverse functions, materials, technologies, and processes across layers [14].

1.2 Floorplanning Challenge of Three Dimensional Circuits

A key challenge in 3D ICs is thermal management. Since silicon dioxide (SiO_2) is a poor thermal conductor, heat is trapped within the layers of the 3D IC, causing a significant increase in temperature [15], [16]. As can be seen in Figure 1.2, the peak temperature on the circuit increased by 53 K, when the ami33 circuit was modeled on a 3D IC rather than a 2D IC. The power density within the 3D structure is beyond the cooling capacity of conventional air-cooled heat sinks [17], [18], and [19]. At high temperatures, the mobility of carriers, such as electrons or holes, decreases [20], thus reducing the current and the frequency of operation. The threshold voltage is also affected by increased temperature, leading to smaller noise margins and reliability concerns. These deleterious effects may lead to performance degradation, technical failure, and increased power dissipation within the IC [21].

Thermal TSVs (TTSVs) can be used to alleviate the thermal issue by acting as heat conduits. However, adding TTSVs to the system limits the available area for devices. The effective area that a TSV would take on a circuit including the TSV keep-out zone (KOZ) is around $35\mu m$ [22]. Moreover, adding TTSVs that exceed 2% of the effective area can cause chip deformation due to a mismatch between the coefficient of thermal expansion (CTE) of the substrate and that of the TTSV. As demonstrated in [23], thermal-aware floorplanning



(a) **(b)**

Figure 1.2: A comparative view of a 2D IC and a 3D IC in terms of on-circuit temperature.

and having a holistic approach to thermal optimization during the design of 3D ICs alleviate the thermal issue. Floorplanning is an early step in the design optimization flow. It is the process of finding coordinates for each block (macro or subchip) included in the IC.

Nonetheless, thermal-aware floorplanning of 3D ICs remains difficult since a thermal term is added to the evaluation of the design. Multi-objective optimization is very challenging given that different metrics are being targeted simultaneously such as area, number of TSVs, and temperature. These objectives often create conflicting optimization goals, leading to trade-offs that are hard to navigate in the solution space. Often, only a subset of these metrics is optimized [24], [25], and [26], ignoring other objectives. Additionally, the solution space of a floorplanning task increases exponentially with each added layer; exploring that space is very time-consuming.

Moreover, dedicated power TSVs are needed to supply and meet the power needs of

devices on upper layers in the 3D stack from power and ground nodes, placed at the bottom layer (layer closest to the package or interposer). During floorplanning, power-dense blocks are generally placed near the heat sink to minimize the thermal impact of the floorplan [23]. However, placing power-dense blocks near the heat sink, far from the package, requires more dedicated power TSVs to supply the power needs of the blocks. While limiting the number of TSVs required in the system, by placing power-hungry devices on lower layers, close to the package and far from the heat sink exacerbates the thermal issue further. Therefore, careful consideration should be given to the co-optimization of the number of TSVs and the junction temperature during the floorplanning process. This issue does not exist in 2D floorplanning since TSVs are not used. Therefore, a 2D floorplanner extended to floorplan in 3D would ignore these key trade-offs and the thermal interactions in the system.

Chapter 2

Background and Related Work

2.1 Metaheuristic Techniques in 3D IC Floorplanning

Most optimization problems are considered nondeterministic polynomial (NP) hard problems [27]. For this reason, several sophisticated optimization algorithms that enable an efficient exploration of the solution space have been developed. These optimization engines are referred to as metaheuristics. Metaheuristics have been developed to find an "optimized" solution rather than an "optimal" one.

2.1.1 Core Concepts and Methods of Metaheuristics

Metaheuristics are approximation methods that can be used for different optimization problems [28]. The two main principles behind any metaheuristic are diversification and

intensification. Examining the behavior and performance of a metaheuristic involves analyzing the strategies employed to ensure a balance between these two concepts. Diversification entails an expansive exploration of the design space, favoring breadth over depth. Rather than thoroughly exploring a region in the solution space, the goal in the diversification stage is to survey many regions. The entropy and randomness in the set of obtained solutions are typically high. Diversification permits localizing good or promising regions. On the other hand, intensification is the process that harvests the knowledge obtained in the diversification stage to exploit the regions selected and investigate them thoroughly. The randomness in the solutions decreases. A balance between these two principles ensures a good solution is obtained in a relatively short time.

There are two major types of metaheuristics: trajectory-based and population-based metaheuristics. Both approaches are iterative. With trajectory-based methods, after each iteration, only one new solution is obtained; however, in population-based methods, several new solutions are obtained at an iteration. In trajectory-based approaches, the starting point, or current solution, is referred to as the transient state. The next state is obtained after a perturbation (or perturbations) and is referred to as the neighbor solution [29]. However, in population-based approaches, the current solutions are referred to as the population.

Parameters set before the optimization process begins are referred to as hyperparameters. They control various aspects of the algorithm's behavior, such as the intensity of diversification and intensification, the rate of convergence, and the selection

criteria for solutions. Choosing suitable hyperparameters is crucial for achieving effective optimization, as they influence the balance between searching broadly across the solution space and refining promising areas in more depth [28].

2.1.2 Applications of Metaheuristics in IC Floorplanning

Several metaheuristics, such as simulated annealing (SA) [30], genetic algorithms (GA) [31], ant colony (AC) [32], and particle swarm optimization (PSO) [33], have been used to floorplan 3D ICs. These approaches use different cost functions, data structures, and models. Moreover, they exhibit a wide range of runtime complexity. The data structure used impacts the quality of the solutions generated and the runtime since the data structure determines which solutions can theoretically be explored. The most popularly used data structures are transitive graphs and sequence pairs [34], [35], and [36]. In [23], a combined bucket and 2D array (CBA) floorplan representation is proposed. The 2D array is composed of transitive closure graphs that store the 2D layouts. The bucket structure stores the 3D relations between blocks. While intralayer perturbations are not constrained, interlayer perturbations are only allowed between two neighboring blocks. This limits the search engine, SA, during the floorplanning process. In this work, area, half-perimeter wirelength (HPWL), TSVs, and peak temperature are optimized. The thermal optimization in [23] is performed by evaluating the temperature within the layer that is farthest from the heat sink. The horizontal and vertical heat paths are disintegrated, and

their impact on temperature is minimized separately. Several optimization procedures were presented in [23] offering trade-offs between runtime and thermal optimization.

In other work, only a subset of performance metrics is typically optimized using various techniques. This usually means that some of the parameters (*e.g.*, temperature) will be significantly worse as compared to a multi-objective optimization. For example, in [26], the footprint of a 3D circuit, the total area of all layers, and HPWL are optimized. Although, temperature gradients and peak temperature are targeted at a later stage, resulting temperatures are significantly higher as compared to [23]. Another example is [25] where area and HPWL are optimized, while the total number of TSVs and peak temperature are only set as constraints.

2.2 Machine Learning Techniques in 3D IC Floorplanning

Learning-based optimization is a rapidly evolving field that combines machine learning and optimization techniques to tackle complex design problems efficiently. In recent developments, artificial intelligence (AI), and more specifically, machine learning (ML), have found application in the design process of 3D ICs.

2.2.1 Core Concepts and Methods of Machine Learning

ML models can be broadly categorized into three main types: supervised learning (SL), unsupervised learning (UL), and reinforcement learning (RL). In SL, the algorithm is trained on a labeled dataset, where input data consists of a set of pairs [37]. In traditional ML, the model is only learning to imitate an expert, and consequently, the machine will never be able to surpass the human. Alternatively, RL shatters this paradigm such that results obtained by the machine can be better than those based on the human expert [38]. It is used in scenarios where there is a need for decision-making over time. Unsupervised learning does not use labeled data for training. Instead, it focuses on finding patterns, clusters, or structures within the data. A special type of unsupervised learning algorithm is the generative adversarial network (GAN). A GAN consists of two neural networks (NN), a generator and a discriminator, which are trained adversarially. The generator attempts to generate data (such as images, text, or other types of content) emulating the real input data, while the discriminator tries to distinguish between the real data and the data generated by the generator. Thus, GAN models are a set of back-to-back ML models where the goal of one model is the opposite goal of the other [39].

2.2.2 Applications of Machine Learning in IC Floorplanning

More recently, an effort to replace non-learning-based metaheuristics in 3D IC design with ML models has been observed. In [40], the use of RL for floorplanning is proposed. The

learning agent learns from previous trials about placements that were beneficial and also about placements to avoid. To assist the agent, feed-forward neural network embedding layers and graph convolutional layers are used to extract features. These features are then passed to the policy network of the RL model. With many runs, the agent learns what strategies and actions should be performed during state S to maximize the reward R. Training the RL model is a lengthy process and does not generalize well. The placement policies learned are only applicable to the specific dimensions of that chip and to the specified set of blocks to place. Moreover, these policies depend on the cost function desired during training. The learning process must restart from scratch if the user adds new parameters to optimize. [40] proposes adding edge-based graph neural network (GNN) layers to facilitate the generalization of the policies learned to other circuits. However, the same cost function and chip dimensions must be maintained. Moreover, it is not possible to optimize area with [40] since the RL agent must place the modules within a predefined space.

As pointed out in [40], floorplanning differs from typical ML problems since learning to floorplan is similar to learning to play "a game with varying pieces, boards, and win conditions" [40]. In this analogy, the pieces are the blocks to place on the circuit, the board is the circuit, and the win conditions are the different cost functions that can be used. This analogy highlights the generalization difficulty encountered by ML models in floorplanning.

Moreover, a clear disadvantage of NNs is the necessity to have a large learning training dataset. In addition, a validation and testing dataset must also be available. Finally,

hyperparameters and the number of training iterations must be carefully chosen. The risk of over-fitting or under-fitting are both relevant in NNs.

Chapter 3

Problem Statement

Although 3D ICs promise to deliver higher performance, reduced power consumption, greater heterogeneity, and a smaller form factor, designing 3D ICs remains extremely challenging. The early stages of 3D IC design, which include thermal optimization and floorplanning, are complex tasks that require careful consideration. Floorplanning 3D ICs becomes more complex with each added layer in the stack, due to the exponential increase in the design space. Moreover, the thermal properties of the circuit and the number of TSVs must be considered during the floorplanning stage to prevent degradation of the IC. The addition of these terms creates many trade-offs in the cost function that are difficult to navigate.

Learning-based methods, in particular ML algorithms, are not often used in optimization problems and suffer from a lack of datasets, vanishing gradients, and poor generalization. Additionally, non-learning-based algorithms for 3D IC floorplanning often are not time-

efficient, as each new solution needs to be evaluated using a complex thermal modeling tool [41].

These challenges for 3D IC design need to be addressed to fully realize the potential of this cutting-edge technology. A thermal modeling, a data structure for floorplanning, and a floorplanning algorithm are presented in this work to address the challenges of the early design of 3D ICs. Our work will be referred to as Matrix Floorplanner [1]. Given a list of blocks with their dimensions, power requirements, and connectivity, this work provides the coordinates of the blocks by simultaneously minimizing area, wirelength, temperature, and number of TSVs in the 3D stack.

Chapter 4

Proposed Matrix Floorplanner

The proposed matrix-based thermal-aware floorplanning algorithm for 3D ICs is described in this section. The matrix-based data structure is P* admissible, as discussed in Section 4.3, and thus supports fast perturbations of the floorplan resulting in a valid floorplan after each change. The data structure, perturbations, and packing are discussed in detail in Sections 4.1 – 4.5. A novel methodology for thermal evaluation during every iteration is described in Section 5.4. The algorithm utilizes the iterative SA engine in which the hyperparameters (such as the cooling schedule), can be tuned by the user.

4.1 Matrix Data Structure

We developed the algorithm named Matrix Floorplanner which exploits matrices to represent valid floorplans. An upper triangular matrix is used to represent the floorplan of each layer

within the 3D structure. The matrix is symmetric of size $m_k \times m_k$, where m_k is the number of blocks on layer k. A floorplan with n layers is represented by using n matrices of variable size (determined by the number of blocks on each layer). The matrix that is used to represent the layer k describes the horizontal and vertical relations among the blocks on the layer, and is denoted M_k. Any two blocks on the same layer are horizontally or vertically related, and have exactly one type of relation. Since blocks cannot be related to themselves, the diagonal of the matrix is also populated with zeros. Blocks **i** and **j** are considered horizontally related, if the right coordinate of **i** (or **j**) denoted x_i^r (or x_j^r) is smaller or equal to the left coordinate of **j** (or **i**) denoted x_j^l (or x_i^l), and they are on the same layer k. The horizontal relation is represented by the value '1' in M_k[i][j] (or M_k[j][i]). Similarly, blocks **i** and **j** are considered vertically related, if the y_i^t (or y_j^t), the top coordinate of **i** (or **j**) is smaller or equal to the y_j^b (or y_i^b), the bottom coordinate of **j** (or **i**), and they are on the same layer k. The vertical relation is represented by the value '0' in M_k[i][j] (or M_k[j][i]). All blocks that meet the coordinate conditions for both horizontal and vertical relations. *i.e.*, diagonally related, are considered only vertically related. This simplification does not limit the representation of floorplans, but rather it supports greater compaction during the packing stage (if applicable).

Furthermore, similar to [34] and [35] in 2D, a single type of relation (horizontal or vertical) between any two blocks is sufficient to maintain P* admissibility, as explained in Section 4.3. A random eight blocks floorplan is depicted in Figure 4.1, note that the asterisks are only shown in the figure for clarity, in the implementation, these cells are

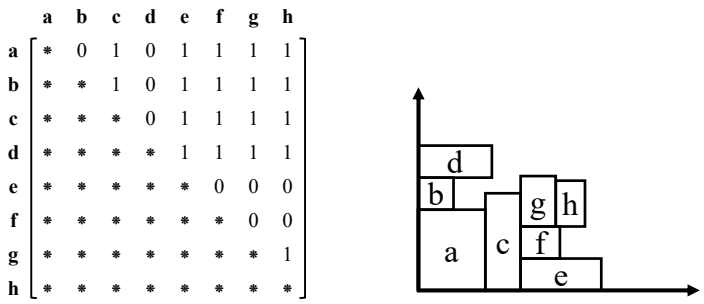
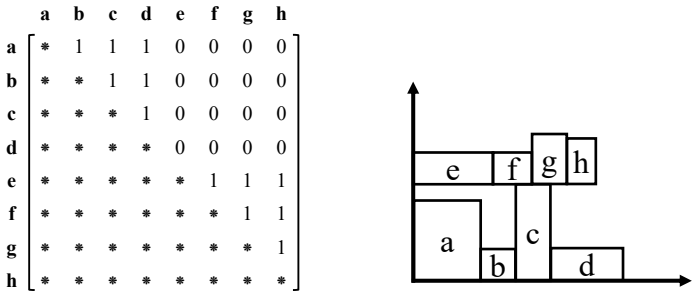


Figure 4.1: An eight-block example floorplan of layer k. (a) Relation matrix M_k, and (b) corresponding layout.

never addressed (contain '-1's). The '1's ('0's) in each row in the matrix in Figure 4.1(a), represent the horizontal (vertical) fan-out of the block listed on the left of that row (points to the properties of each block). For example, in row **b** of the matrix M_k in Figure 4.1(a), '1's appear for blocks (columns) **c**, **e**, **f**, **g**, and **h**, denoting that these blocks are horizontally related to block **b**, *i.e.*, located to the right of block **b**. Similarly, in row **e** of M_k, '0's appear for blocks (columns) **f**, **g**, and **h**, denoting that these blocks are vertically related to block **e**, *i.e.*, located above block **e**.

4.2 Matrices Initialization

The initial floorplan, prior to the start of the optimization process, can significantly affect the performance of the algorithm. Initialization of the matrices is, therefore, a crucial step in the algorithm. If the matrices are randomly initialized, the floorplanner may never converge to a good result. Alternatively, a complex matrix initialization technique impedes runtime



(a) **(b)**

Figure 4.2: An eight-block example of a row-initialized floorplan of layer k. (a) Relation matrix M_k, and (b) corresponding layout.

and may lead to a local minimum.

The first step in the initialization process of the Matrix Floorplanner is to assign blocks to a certain layer. The number of layers in the design is defined by the user and will not change during optimization. The allocation of blocks to each layer is performed randomly and during the optimization process, blocks may move across layers. Once blocks are allocated to a certain layer, the corresponding matrix is created.

The matrix of each layer is populated in a directed manner to form rows of blocks. The row-based initialization is depicted in Figure 4.2. Any block within the same row is horizontally related to all other blocks in the same row, denoted with '1's in the corresponding cells of M_k. Any two blocks in different rows are considered vertically related and denoted by '0's in the corresponding cells of M_k. To keep the aspect ratio (AR) of the width over the height of the initial floorplan close to 1, the number of blocks in each row is limited. This matrix initialization has a runtime complexity of $O(m^2)$, where $m = \max\{m_k\}$ for

$k = 0, ..., n-1$ (n is the total number of layers), *i.e.,* m is the number of blocks in the most populous layer.

4.3 P* Admissibility

P* admissibility is a metric to compare non-slicing floorplanning algorithms [35]. This metric guarantees the fast generation of valid floorplans and is a desired property for floorplanning algorithms. P* admissibility includes the following five conditions [35].

- The design space of the floorplanner must be finite. This condition is intrinsically satisfied for every floorplanner, and also for the proposed matrix algorithm, as there is a finite, albeit large, number of ways to place blocks within a practically limited outline.

- The produced solution should be feasible. The Matrix Floorplanner satisfies this condition since the packing step (described in Section 4.5) ensures that no overlap among blocks is permitted.

- The runtime complexity of the packing and cost computation functions, must be at most polynomial. This condition is also met by the Matrix Floorplanner, since both the packing and cost computation functions exhibit a runtime complexity of $O(m^2)$, as described in, respectively, Section 4.5 and Chapter 5.

- The lowest cost solution that is generated by the algorithm, must represent an optimal physical placement of the blocks, *i.e.*, all possible placement solutions (including non-slicing placements) must be accessible to the algorithm and could be theoretically explored. This condition is also met by the Matrix Floorplanner since no feasible solution is excluded from the solution space, in other words, the matrix algorithm is non-slicing and allows for any valid placement of blocks [35]. The perturbations utilized in the Matrix Floorplanner support the exploration of the entire design space as blocks can not only move within a layer but also among layers.

- The geometric relation between any two blocks must be defined in the representation. The Matrix Floorplanner is designed to accommodate this condition, as the relative position of any two blocks within the netlist, is stored in the relation matrix M_k. Similar to TCG [34], SP [36], and TCG-S [35], only one type of relation between a pair of blocks is sufficient to fulfill this criterion. For example, in TCG-S, two blocks cannot be related in both the horizontal and vertical transitive closure graphs.

The proposed Matrix Floorplanner is P* admissible as it satisfies all of the conditions for P* admissibility. Proof that the runtime complexity of the perturbations, packing, and cost computation functions, is polynomial, is provided in the following sections.

4.4 Perturbations

As part of the iterative optimization process, the floorplan of the 3D ICs is perturbed to generate a new floorplan with a new cost, enabling exploration of the solution space. To maintain P* admissibility (as explained in Section 4.3), all perturbations must be of polynomial runtime complexity, and all floorplans produced after perturbations must be valid (no overlapping blocks).

Seven different perturbations are performed as part of the Matrix Floorplanner. These perturbations either affect one layer (intralayer perturbations) or two layers (interlayer perturbations) within the 3D IC. For all perturbations, the layer(s) are chosen randomly. All perturbations are explained on the small eight-block netlist that was initialized as in Figure 4.2. The resulting floorplan (after packing) for each perturbation, is shown in Figure 4.3.

- **Rotation (intralayer).** Block **d** is randomly selected and rotated. The height and the width of the block are switched. This perturbation does not change the relation matrix, since the geometric relations between **d** and the rest of the blocks on the layer are not impacted by the change in dimensions of **d**. The packing function will make sure that blocks move to accommodate the new dimensions of **d** and ensure a valid floorplan. This perturbation, therefore, is performed at a constant runtime complexity O(1).

- **Aspect ratio (intralayer).** This perturbation is only applicable to soft blocks, *i.e.*,

blocks for which only the area is provided (not fixed width and height). During this perturbation, a random soft block **c** is selected. The AR of **c** will be randomly chosen from a range that is specified by the user, and the new width and height of block **c** are calculated according to $\mathbf{c}_{\text{height}} = \sqrt{\mathbf{c}_{\text{area}} \cdot \text{AR}^{-1}}$ and $\mathbf{c}_{\text{width}} = \mathbf{c}_{\text{area}} \cdot \mathbf{c}_{\text{height}}^{-1}$. This perturbation has a runtime complexity of O(1) since the only required change is in the properties of the block.

- **Swap (intralayer).** Blocks **d** and **f** (from the same layer k) are randomly selected and the coordinates are swapped. Block **d** inherits all geometric relations of **f** and vice versa. The relation matrix M_k is updated by switching the order of the pointers to the properties of the blocks (name, dimensions, connectivity, power, etc) in the rows and columns. This operation maintains the validity of M_k, while only changing the properties of the blocks. The runtime complexity of the swap perturbation is, therefore, O(1).

- **Geometric relation flip (intralayer).** Two blocks **a** and **d** (from a same layer k) are randomly selected and their relative geometric relation with respect to one another is flipped. Previously, **d** was vertically related to **a** and after the geometric relation flip perturbation, **d** is horizontally related to **a**. Since only one value in M_k has to be changed (from '0' to '1'), the runtime complexity of this operation is O(1).

- **Geometric relation flip and swap (intralayer).** This perturbation effectively

consists of a geometric relation flip perturbation followed by a swap perturbation (performed on the same pair of blocks). In Figure 4.3(e), blocks **g** and **a** are perturbed. The runtime complexity of this operation is O(1) since the largest runtime complexity of its constituents is O(1).

- **3D swap (interlayer).** Unlike in intralayer swap, two blocks, **e** and **i**, are randomly selected from *different* layers (k and l). Note that only layer k is shown in Figure 4.3(f). These blocks will switch layers and inherit the relative geometric relations from each other. This perturbation maintains the validity of M_k and M_l, while only changing the properties of the blocks. This operation is performed with a runtime complexity of O(1).

- **3D move (interlayer).** Block **j** is randomly selected from source layer l (not shown in Figure 4.3(g)) and moved to destination layer k. Unlike other perturbations, 3D move changes the number of blocks that was allocated to each layer during the initialization phase. If a layer contains only one block, it will never be selected as a source layer for the 3D move perturbation. This operation is performed with a runtime complexity of O(m), since removing a block from layer l entails adding and filling a column in M_k (the relation matrix of the destination layer). Block j is always placed to the right (*i.e.*, in a horizontal relation) of all other blocks on layer k.

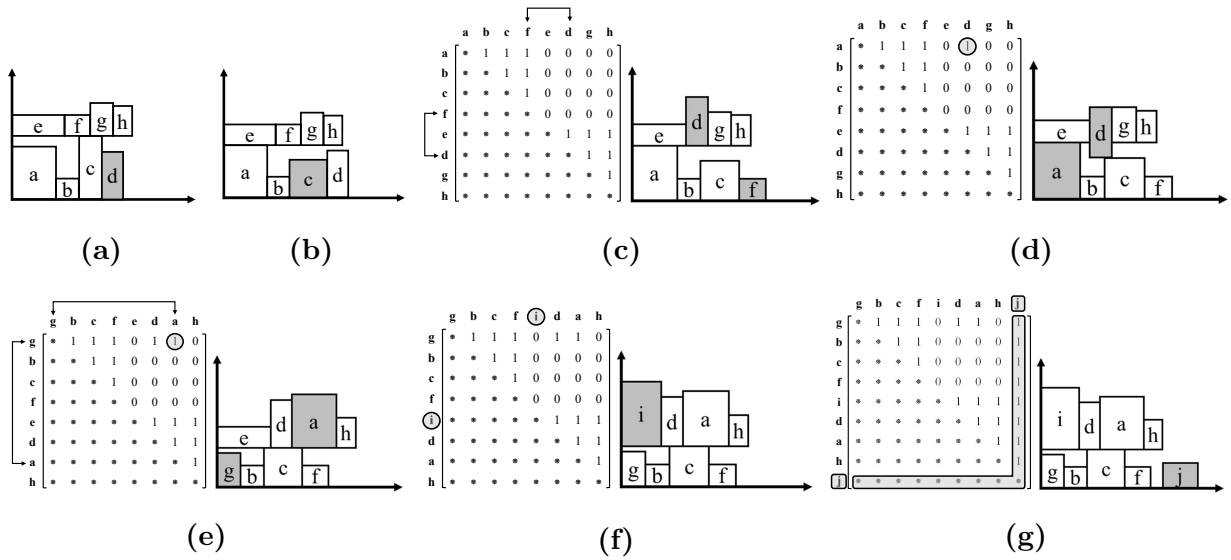


Figure 4.3: Perturbations of the Matrix Floorplanner. Affected blocks and respective changes in the relation matrix are grey-shaded. (a) Rotation of **d**. (b) Change in the AR of **c**. (c) Swap between **d** and **f**. (d) Geometric relation flip between **a** and **d**. (e) Geometric relation flip followed by swap between **a** and **g**. (f) 3D (interlayer) swap between **e** (moved from shown layer to a different, not shown, layer) and **i** (moved from different, not shown layer, to shown layer). (g) 3D (interlayer) move of **h** (from different, not shown layer, to shown layer).

4.5 Packing

The packing function determines the coordinates of all blocks in the netlist, based on the information stored in the set of relation matrices (across the 3D IC). The packing step must always follow any performed perturbations, since each perturbation is likely to impact the coordinates of all blocks on the layer. For example, as shown in Figure 4.3(b), a seemingly small change in the AR of **c**, affects the coordinates of blocks **e**, **f**, **g**, and **h**. The perturbation function does not adjust the coordinates of all the impacted blocks; it is the job of the packing function. It is not required to perform packing, however, after each perturbation.

The runtime complexity of the packing function is in $O(m^2)$. In practice, packing is performed for each layer of the 3D IC, but since the number of layers n tends to be small ($n < 10$) in any practical design, it can be accurately approximated as $O(m^2)$. To determine the coordinates of the blocks on layer k, the packing function parses through the entire M_k. To explain the packing process, an example is provided in Figure 4.4 which performs packing on the same relation matrix as the one obtained in Figure 4.3(g). Packing of block **a** (step 7) is highlighted. In this example, block **a** is on the right of blocks **g**, **i**, and **d** ('1's in the matrix). Thus, the left coordinate of **a** is $x_a^l = \max\{x_g^r, x_i^r, x_d^r\} = x_d^r$. Similarly, **a** is above blocks **b**, **c**, and **f** ('0's in the matrix). Thus, the bottom coordinate of **a** is $y_a^b = \max\{y_b^t, y_c^t, y_f^t\} = y_c^t$.

The described packing scheme ensures that no overlaps occur since each newly placed block is either to the right or above all other previously placed blocks in the layer. This is

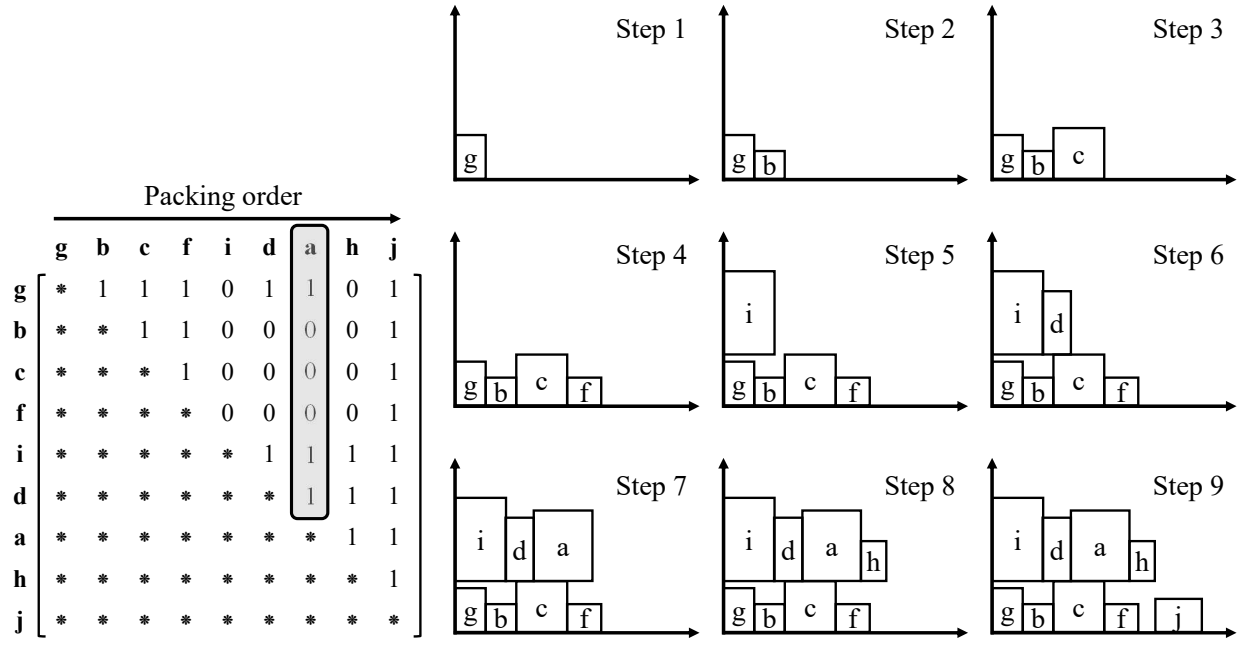


Figure 4.4: An example of a packing procedure of the solution obtained in Figure 4.3.(g). The same matrix is provided for convenience. An example of placement of block **a** (step 7) is highlighted in the matrix.

guaranteed by the requirement that all blocks be either horizontally or vertically related.

Chapter 5

Computation of Performance Metrics

SA is used as a multi-objective optimization engine. Several floorplan performance metrics are combined using a weighted sum into a single cost parameter,

$$C = \frac{w_{\text{area}}}{n_{\text{area}}} \cdot A + \frac{w_{\text{wl}}}{n_{\text{wl}}} \cdot wl + \frac{w_{\text{temp}}}{n_{\text{temp}}} \cdot T_{\text{L}} + \frac{w_{\text{TSVs}}}{n_{\text{TSVs}}} \cdot N_{\text{TSVs}} \tag{5.1}$$

where, C, A, wl, T_{L}, and N_{TSVs} are the, respectively, cost, area (of the largest layer), total wirelength, thermal load, and total number of TSVs within the 3D IC. w and n are the, respectively, weight and normalization parameters for each of the performance metrics. The weights are used to scale the metrics according to their importance in a specific netlist, such that the SA engine will focus on optimizing the important metrics. The normalization parameters ensure that all metrics are evaluated on the same scale of values. Both w and n of each metric can be controlled by the user to tune for a specific netlist. Each metric is

described in the following subsections.

5.1 Computation of the Area

During optimization, the area of the largest layer within the 3D IC is included in the cost function. The largest layer varies throughout the SA process; at every iteration, a different layer may exhibit the worst (largest) area. This ensures that all layers are targeted and improved. It is not required to optimize the sum of the area of all layers in 3D ICs since technological limitations do not support layers of different areas within a single 3D structure. Moreover, for area-focused floorplanning, the largest layer may be directly targeted during the layer selection step in all perturbations.

The AR of each layer is constrained within the floorplanning algorithm to control the footprint of the design. A certain footprint (*e.g.*, rectangle with high/low AR) can be also targeted. The runtime complexity of area computation is O(m) (assuming that n is small, since, in practice, 3D ICs are up to five layers only).

5.2 Computation of the Wirelength

Wirelength is computed as the HPWL of the bounding box among all blocks that are connected to the same wire [42]. Two types of connections are typically described in a netlist – blocks connected to other blocks, and blocks connected to external terminals. The

computation of block-to-block wirelength is performed by identifying the (x,y) coordinates of all blocks connected to the same wire. For blocks that are located on different layers, the HPWL is calculated for the smallest bounding box of the projections of all blocks to the same plane, in other words, the z coordinate of the blocks is ignored as if they were all on the same layer. In this case, TSVs are ignored in the computation of wirelength, since the length of the TSVs is typically not provided at this stage. Alternatively, if TSV dimensions are available, they can be included in this computation. Nonetheless, TSVs are accounted for in the optimization of the number of TSVs, as described in Section 5.3.

The connectivity of blocks to external terminals is further divided into two groups, a terminal connects to a single block or a terminal connects to multiple blocks. In the former case, the terminal is assumed to be attached to the block and, therefore, the length of the connecting wire is assumed to be zero. In the latter case, the terminal is assumed to be placed in the location of the geometric center of the connected blocks on the bottom layer of the 3D IC (the layer that is closest to the connection to the package). The HPWL is computed for every block connected to the terminal, using the same projection approach. The runtime complexity of the computation of wirelength at each iteration of the SA, is O(m^2).

5.3 Computation of the Number of TSVs

Optimization of TSVs is important in 3D ICs since TSVs add reliability concerns to the system. Differences in the coefficient of thermal expansion between TSV material (typically

copper) and the substrate material (typically silicon), may lead to physical damage of the platform [43]. TSVs also pose electrical reliability concerns, for example, noise coupling [43]. For two blocks in the floorplan, located on layers l_1 and l_2, the number of TSVs between the blocks equals $|l_1 - l_2| \cdot n$, where n is the number of connections between the two blocks. This approach is used to determine the total number of TSVs in the floorplan.

It is reasonable to assume that all terminals of the 3D ICs are located in the bottom layer of the stack. For every block connected to a terminal, therefore, the number of TSVs that are required for the connection is calculated and added to N_{TSVs}. The runtime complexity of the computation of the number of TSVs in the floorplan at each iteration of the SA, is $O(m^2)$.

5.4 Computation of the Temperature

Heat is a significant challenge in 3D ICs, since it is trapped with the layer of SiO_2 of the 3D stack, causing a plethora of issues, including a degradation in performance, mechanical stress, and a decreased lifespan. For these reasons, estimating, predicting, and optimizing the temperature is of paramount importance in 3D ICs. The proposed work regarding the thermal issue would center around these three main topics.

Thermal-aware floorplanning alleviates the thermal issue of 3D circuits [23]. It entails thermally evaluating floorplans during the floorplanning process and rejecting floorplans with high temperatures. The number of iterations required to obtain a high-quality solution can be very large, especially for large netlists. Moreover, thermal evaluation dominates

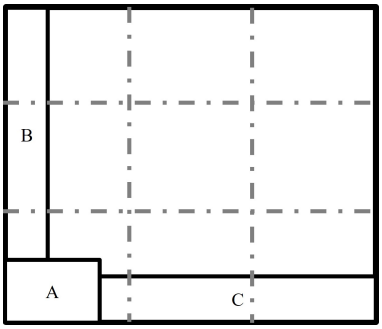


Figure 5.1: A layer of the 3D stack divided into thermal cells

the runtime of an iteration of floorplanning. Traditional thermal evaluators of floorplans are inefficient. The floorplanner optimizes peak temperature during every iteration of the simulated annealing process. As the number of iterations required to obtain a high-quality solution can be very large, especially for netlists with a significant number of blocks, a fast thermal evaluation procedure was developed. The fast thermal evaluation speeds up the runtime without impeding the quality of the solution. This evaluator was used in [1].

In this procedure, the peak temperature of the 3D IC is estimated by considering three main factors: the power density, the number of TSVs, and the distance to the heat sink. The power density represents an estimation of the power dissipated within a region of the IC. The distance from the heat sink estimates how efficiently heat is being dissipated by the sink. Finally, the number of TSVs is also considered since TSVs act as thermal conduits, that either dissipate heat away from a region in the IC towards the heat sink or bring in heat from the lower layers of the 3D stack towards a region in the stack. Each layer of the 3D IC is divided into a thermal grid, as can be seen in Figure 5.1. The size of each cell within

the thermal grid is determined according to the user-controlled number of cells N_c and the area of each layer. In the next step, the power dissipated within each cell of the thermal grid is determined. The power P_c dissipated in cell c is calculated from the power dissipated within the blocks overlapping with cell c, prorated according to the ratio of overlap. For example, if three blocks **x**, **y**, and **z** are overlapping with cell c by, respectively, r_x, r_y, and r_z (ratio of overlap), and the power of the blocks is, respectively, p_x, p_y, and p_z, then the power dissipated in cell c is $P_c = p_x \cdot r_x + p_y \cdot r_y + p_z \cdot r_z$. The thermal load T_{L} that is included (minimized) in the overall cost function is, therefore

$$T_{\mathrm{L}} = \max\{P_c \cdot d_c \cdot \frac{N_{\mathrm{TSVs}}^{n-1} + 1}{N_{\mathrm{TSVs}}^{n} - 1}\} \tag{5.2}$$

where, d_c is the distance of cell c to the heat sink, N_{TSVs}^{n} is the number of TSVs leaving the cell (toward the heat sink), *i.e.*, assist in removing heat from the cell, and N_{TSVs}^{n-1} is the number of TSVs connecting to the cell from the previous layer, *i.e.*, adding heat to the cell from other cells (farther away from the heat sink). The runtime complexity of the computation of the thermal cost at each iteration of the SA, is $\mathrm{O}(m \cdot N_c)$. Note that from experiments, N_c can be a relatively small number (similar to the number of blocks or smaller) to obtain good thermal optimization. The runtime complexity can, therefore, be accurately estimated as $\mathrm{O}(m^2)$.

This thermal modeling is used during the floorplanning iterations since, during

floorplanning, many thermal evaluations are required to assess and compare different solutions. After completion of the floorplanning stage, the final floorplan can be evaluated using a FEM tool such as COMSOL [44] or other thermal modeling tools such as ARTSim [2].

Chapter 6

A Comparative Analysis and Performance Evaluation

This chapter presents a comprehensive analysis and performance evaluation of the Matrix Floorplanner on different netlists. The first part of this chapter delves into the comparative aspects, highlighting how the Matrix Floorplanner distinguishes itself from existing P* admissible floorplanners, particularly in its ability to effectively handle 3D ICs challenges (large design space and complex evaluation). In the subsequent section, the Matrix Floorplanner's practical performance is emphasized using standard benchmarks and industrial netlists, showcasing its superiority in optimizing key metrics such as area, wirelength, TSV count, and peak temperature, all while ensuring a reasonable runtime.

6.1 Comparative Analysis

The matrix-based representation presented is similar to the transitive-closure-graph (TCG) approach in [34], and [35], since all three are P* admissible. [34], and [35] are among the most popular graph-based representations used in floorplanning. However, the matrix floorplanner distinguishes itself from these existing graph-based floorplanners due to several key advantages.

First, the matrix-based floorplanner is used to floorplan 3D ICs, contrary to TCGs [34], and [35], which can only floorplan and represent 2D ICs. Moreover, the Matrix Floorplanner eliminates the redundancy created by existing graph approaches in 3D. For example, in [45] constraints on the z-dimension are not considered. Blocks are not given discrete coordinates in the z-dimension which leads to unfeasible floorplans (if no evaluation step follows) and creates a larger and more complex solution space to explore.

Furthermore, existing state-of-the-art graph-based approaches, such as the [23], do not explore the solution space in 3D in an efficient manner. The intralayer perturbations permissible in [23] are identical to the perturbation of [35], since [23] uses TCGs to represent the 2D floorplans, creating a TCG for each layer of the 3D stack. However, the interlayer perturbations, in particular inter-layer move, are limited to blocks that are neighbors across the 3D layers (similar x, y coordinates, but different z). Therefore, the bucket approach limits and weakens the diversification phase in the exploration of the solution space. The CBA-based of [23] would require more iterations and a longer runtime

to reach a solution easily accessible to the matrix-based floorplanner. Moreover, the condition of P* admissibility is that each permissible solution is accessible to the floorplanner. By limiting the perturbations, the P* admissibility of the [23] does not hold.

[46] uses B* trees to represent 3D floorplans. Yet, this data structure is also not P* admissible the feasibility of the proposed solutions is also not guaranteed. During the packing stage, a block's intended location might be used by another block previously placed causing overlaps. In addition, the geometric relation between any two modules is not defined in the representation.

As summarized in Table 6.1 the Matrix Floorplanner distinguishes itself from other P* admissible and non P* admissible floorplanning algorithms by allowing a broader range of perturbations. These perturbations have a more significant impact on the layout, which is important during the diversification stage of the SA engine. Moreover, the Matrix Floorplanner, unlike other floorplanners, does not limit the types of perturbations in 3D to accessible neighboring blocks. In addition, the Matrix Floorplanner simultaneously optimizes four important metrics for 3D ICs: area, wirelength, number of TSVs, and peak temperature, while maintaining acceptable runtime (detailed in Section 6.2). While, due to the complexity of the simultaneous optimization of all mentioned metrics, many other floorplanning algorithms constrain some metrics while optimizing only a subset.

Paper	Constraint	Optimized metrics	Data Structure	Drawbacks
[23]	Peak temperature	Area, wire length, TSVs	CBA based on [35] limiting perturbations in 3D	Incomplete temperature estimation (one direction only)
[47]	Peak temperature	Area, TSVs	Tile-based floorplans (not P* admissible)	Relies on carbon nano-tubes to cool down circuit
[46]	Peak temperature	Area, wire length	B* Tree with no z-direction perturbations	No 3D perturbations, layer assignment overhead
[45]	N.A.	Area, wire length	TCG-S	Non-discreet z-coordinates
[40]	Area	Wirelength, congestion density	Vectors	No temperature reduction, Poor generalization

Table 6.1: Various state-of-art floorplanners, their cost functions, features, and drawbacks.

6.2 Simulation Results

The proposed Matrix Floorplanner is evaluated on standard MCNC and GSRC benchmarks obtained from [48]. The experiments were performed on a Lenovo machine, Intel Core i5 (four cores) processor, at a frequency of 1.6 GHz. As part of the evaluation, area 5.1, wirelength 5.2, thermal load 5.4, number of TSVs 5.3, and runtime, were obtained. To get an accurate result for the peak temperature, the final floorplan was evaluated using a HotSpot thermal model [41] that was adapted to accommodate non-slicing floorplans. The results of the evaluation are compared to previously proposed thermal-aware floorplanners [23] and [49]

and the results are listed in Table 6.2 – 6.6.

	CBA-T [23]	Li [49]	Matrix Floorplanner [1]
ami33	$4.14 \cdot 10^{-7}$	$3.61 \cdot 10^{-7}$	$3.97 \cdot 10^{-7}$
ami49	$1.84 \cdot 10^{-5}$	$1.24 \cdot 10^{-5}$	$1.38 \cdot 10^{-5}$
n100	$6.56 \cdot 10^{-8}$	$5.74 \cdot 10^{-8}$	$5.90 \cdot 10^{-8}$
n200	$6.91 \cdot 10^{-8}$	$6.83 \cdot 10^{-8}$	$5.61 \cdot 10^{-8}$
n300	$1.06 \cdot 10^{-7}$	$4.14 \cdot 10^{-7}$	$9.65 \cdot 10^{-8}$

Table 6.2: Results of the evaluation of area (m^2) of the Matrix Floorplanner and comparison to previous work, for a 4-layer 3D IC.

	CBA-T [23]	Li [49]	Matrix Floorplanner [1]
ami33	0.024442	0.033111	0.024234
ami49	0.477646	0.198711	0.392008
n100	0.092456	0.072622	0.092008
n200	0.190886	0.179565	0.206000
n300	0.253837	0.257049	0.255564

Table 6.3: Results of the evaluation of wirelength (m) of the Matrix Floorplanner and comparison to previous work, for a 4-layer 3D IC.

	CBA-T [23]	Li [49]	Matrix Floorplanner [1]
ami33	98	N/A	81
ami49	211	N/A	242
n100	1044	N/A	1207
n200	2021	N/A	1984
n300	2261	N/A	2144

Table 6.4: Results of the evaluation of the number of TSVs of the Matrix Floorplanner and comparison to previous work, for a 4-layer 3D IC.

To accommodate for the disparity in thermal modeling of the cooling systems across floorplans, an effort was made to match the thermal parameters in [23]. Specifically, matching

	CBA-T [23]	Li [49]	Matrix Floorplanner [1]
ami33	160	338	144
ami49	151	342	134
n100	158	273	130
n200	156	237	129
n300	167	270	113

Table 6.5: Results of the evaluation of peak temperature (°C) of the Matrix Floorplanner and comparison to previous work, for a 4-layer 3D IC.

	CBA-T [23]	Li [49]	Matrix Floorplanner [1]
ami33	466	1.6	144
ami49	521	6.1	114
n100	4322	21	451
n200	6843	86	1035
n300	17484	256	1957

Table 6.6: Results of the evaluation of runtime (s) of the Matrix Floorplanner and comparison to previous work, for a 4-layer 3D IC.

the thermal parameters such that similar temperatures are obtained while floorplanning benchmarks without any thermal optimization. Furthermore, an FEM simulation (using COMSOL) was performed on a two-layer ami33 netlist and the thermal parameters (*e.g.*, dimensions of heat sink and heat spreader) of the last computation of temperature (after obtaining the final floorplan) were tuned accordingly. The temperature results obtained from the thermal model of the Matrix Floorplanner were within 3% of the values obtained from the FEM simulation.

It can be observed from the results in Tables 6.2, 6.5, and 6.6 that the proposed Matrix Floorplanner produces improved solutions as compared to [23], in area, temperature, and

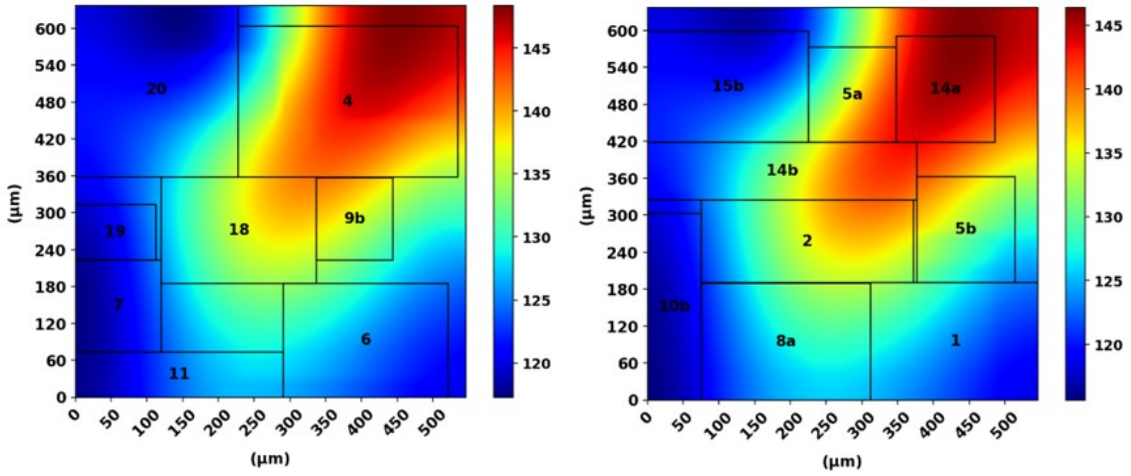


Figure 6.1: Thermal map visualization for the optimized layout of ami33 on a four-layer 3D IC; (a) layer 1 and (b) layer 4.

runtime. The algorithm produces comparable results in terms of TSVs and wirelength. Note, that additional results are provided in [23] (for CBA-T-Fast and CBA-T-Hybrid), however, these results exhibit significantly higher peak temperatures and therefore were not chosen for comparison in this work. Nonetheless, the proposed Matrix Floorplanner also improves on area and runtime (in addition to peak temperature), as compared to these algorithms.

Comparing the obtained results to [49], it can be observed that the Matrix Floorplanner significantly outperforms with respect to temperature, due to their simple thermal model. It is also interesting to note, that although the work in [49] exhibits superior area and wirelength for small floorplans, the Matrix Floorplanner produces better results for large netlists, in area (for n200 and n300) and in wirelength (for n300). Also, the work in [49] does not target the optimization of TSVs.

To support convenient analysis of the resulting floorplans, a thermal visualization tool has been used [2]. The generated thermal maps of the four layers ami33 netlist (MCNC benchmark) are illustrated in Figure 6.1. The generated thermal maps of the floorplan in Figure 6.1 verify the thermal feasibility of the generated floorplan.

In addition to the standard MCNC and GSRC benchmarks, a few netlists from industrial partners were obtained and then floorplanned and evaluated using the Matrix Floorplanner. Listed in Table 6.7, are the simulation results of three netlists consisting of 10, 30, and 50 soft blocks. Since the blocks are soft, the AR perturbation is activated during the optimization of the floorplan. Standard thermal cooling parameters (similar to the ones used in HotSpot [41]) were used and the size of the heat sink and heat spreader were adapted according to the size of the floorplan. Note that the sum of the area of all blocks without any white space is also listed in Table 6.7 and is used as the theoretical minimum. The percent of additional white space in the floorplans that are generated using the Matrix Floorplanner as compared to the theoretical minimum is also listed.

A thermal visualization tool was also used for the netlists obtained from industrial partners. The generated floorplan of the 30 blocks industrial netlist on two layers (from Table 6.7) is shown in Figure 6.2. Visual inspection of the floorplan in Figure 6.2 verifies that a high-quality solution was produced in terms of area (few white spaces are visible). This result is confirmed by the low percentage of white space in this benchmark (3.8%).

	Layers	Area w/o white space (m^2)	Area (m^2)	Percent of white space	WL (m)	TSVs	T (C)	Time (s)
10 blocks	2	$2.89 \cdot 10^{-5}$	$3.09 \cdot 10^{-5}$	6.9%	0.064178	11	76	15
30 blocks	2	$7.76 \cdot 10^{-5}$	$8.06 \cdot 10^{-5}$	3.8%	0.535507	56	94	168
30 blocks	4	$3.88 \cdot 10^{-5}$	$4.39 \cdot 10^{-5}$	13.1%	0.359770	95	133	68
50 blocks	4	$6.40 \cdot 10^{-5}$	$7.57 \cdot 10^{-5}$	18.3%	0.924614	206	135	67

Table 6.7: Results of the evaluation of the proposed Matrix Floorplanner on netlists obtained from industrial partners.

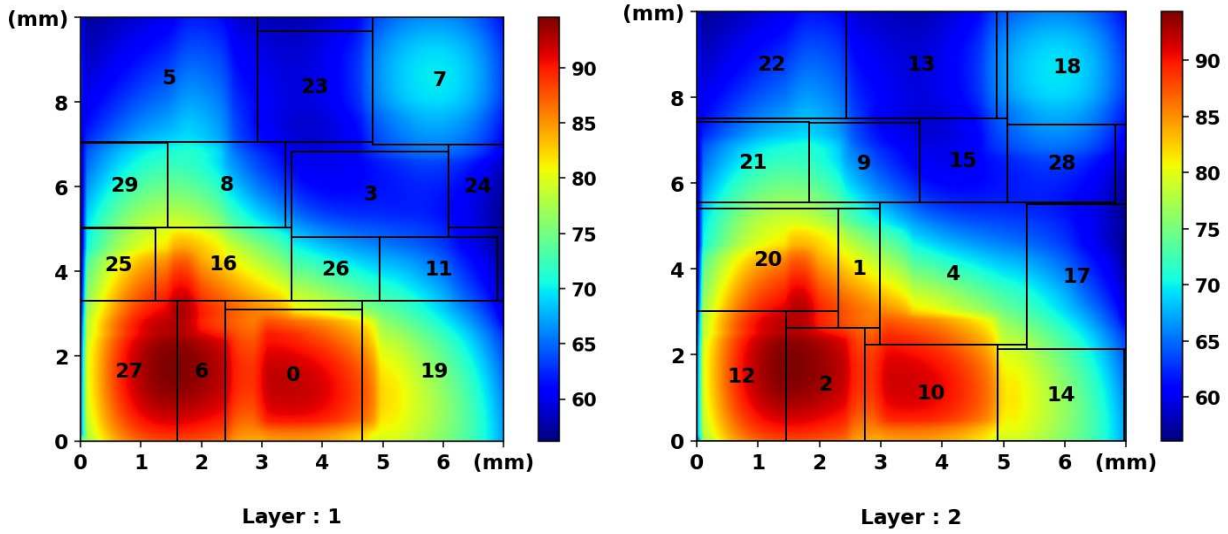


Figure 6.2: Thermal map visualization of a two-layer 30 blocks netlist obtained from an industrial partner. Numbers on the map represent block names.

Chapter 7

Future work

A main challenge with non-learning-based approaches is the difficulty of selecting hyperparameters. There are many hyperparameters in popular optimization metaheuristics that are difficult to choose, for example, 27 HPs are used in the Matrix Floorplanner. These hyperparameters can be, for example, the number of iterations, the starting temperature of the simulated annealing process, the rate of acceptance of the bad solution, and the probability of obtaining a particular perturbation at a certain temperature. A human expert cannot test all combinations possible. These hyperparameters significantly impact the quality of the floorplan obtained after optimization. A badly tuned floorplanner can produce floorplans with considerably worse cost than a well-tuned one. Finally, these hyperparameters are intrinsically dependent on the input netlist. To mitigate this issue, many floorplanning iterations are required. [50] explains that "weeks of iterations" are

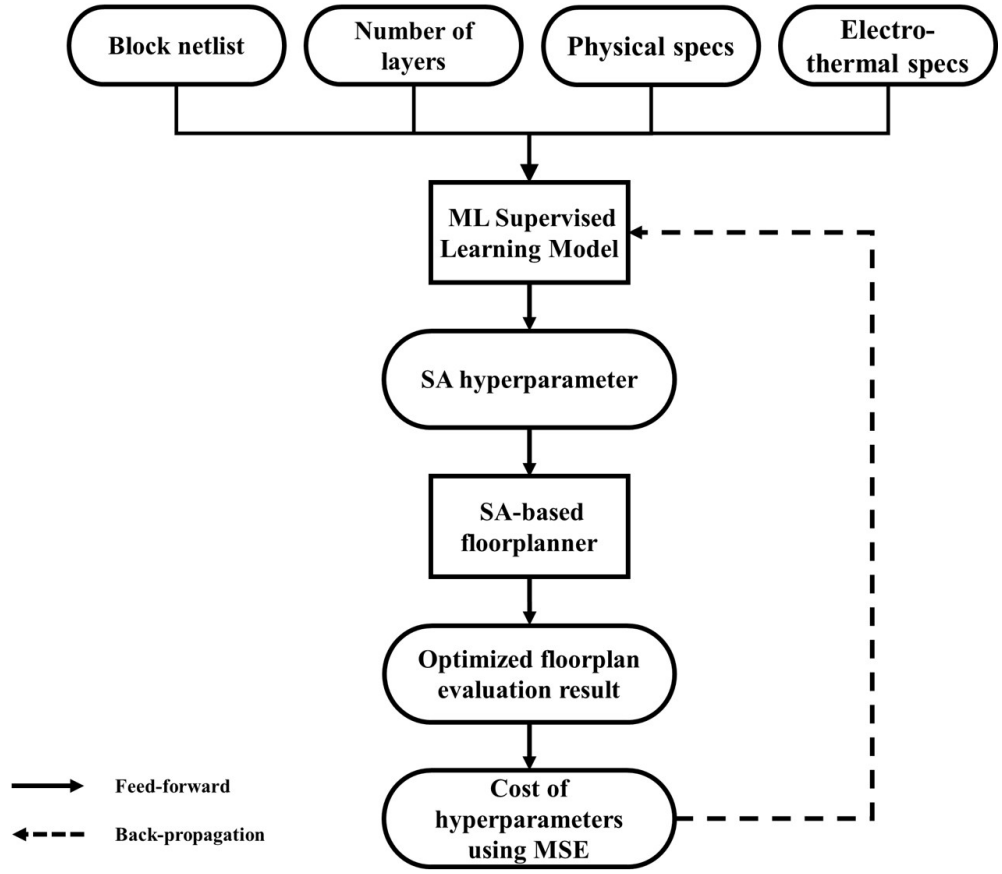


Figure 7.1: Proposed training of the SA hyperparameter optimization model.

needed to select the hyperparameters to use.

Therefore, we propose an ML model that would determine what hyperparameters to use to alleviate the task of the human expert in testing and selecting the hyperparameters. As shown in Figure 7.1, the input of the ML model is the complete netlist of the circuit to floorplan. The output of the model is the set of hyperparameters to use to obtain an optimized floorplan using a specific engine. The output of the ML model is the entire set of hyperparameters required to run the SA engine for this netlist. A novel technique to

facilitate the selection of hyperparameters of the SA engine for different netlists is proposed. The proposed technique is based on reinforcement learning (RL) policy network. The RL network is trained on many types of netlists to determine the number of iterations, starting temperature, and cooling rate of the SA engine. The training dataset of the RL network consists of different circuit netlists with their blocks, dimensions, connectivity lists, and terminal pins. The RL network is trained using Corblivar floorplanning tool [51]. A custom reward function was developed for the training phase to reflect the performance of the SA engine with respect to the hyperparameters. For a netlist, the ideal solution is a weighted sum of the ideal area and wirelength. The ideal area represents the sum of each block's area. Meanwhile, the ideal wirelength is determined by doubling the square root of the ideal area, reflecting the half perimeter wirelength. The performance of the SA engine with the pre-trained hyperparameters on the validation data is within 5% of the state-of-the-art, with respect to area.

$$L = \frac{Area - Area_{Ideal}}{Area_{Ideal}} + \frac{WL - WL_{Ideal}}{WLIdeal} + \frac{Temp - TempIdeal}{Temp_{Ideal}} \tag{7.1}$$

This loss function allows to measure the distance between the ideal solution and the solution obtained using these hyperparameters. The dataset used for training was developed to resemble "real-world netlists". The netlists have between 10 to 350 blocks. The dimensions of the blocks are within the range of blocks from ami33 and n300 benchmarks, as well as the same range for the aspect ratios of blocks. GSRC and MCNC benchmarks will be reserved

for testing purposes.

Chapter 8

Conclusion

In conclusion, 3D ICs are a promising solution to the escalating computing demands of modern applications. They allow more heterogeneity in the circuit and reduce interconnect delay. However, the transition to 3D ICs introduces its own set of challenges, particularly in the realm of design optimization. Most optimization tools were designed for 2D ICs and do not consider the unique challenges posed by 3D ICs in terms of thermal management and floorplanning.

To address these challenges and improve the temperature optimization of a circuit, a fast thermal evaluator was developed and published. This tool can be used when an iterative process requires the modeling and thermal evaluations of many floorplans. This evaluator is based on the unique thermal properties of 3D ICs including the behavior of TSVs as heat conduits. This evaluator also uses the individual properties of the blocks and their

coordinates on the 3D circuit.

Moreover, a novel SA-based floorplanner named the "Matrix Floorplanner" was completed. This floorplanner outperforms state-of-the-art floorplanners and meets the highest standard for floorplanning algorithms, P* admissibility.

The algorithm relies on the relative horizontal and vertical relationships among the blocks within each layer of the 3D structure. These relationships are utilized by the data structure to facilitate rapid manipulations and modifications to the layout during the iterative process. The use of matrices for executing perturbations ensures that the resulting floorplans remain feasible and free of overlaps. Additionally, the impacts of both intra-layer and inter-layer perturbations are thoroughly examined.

The overall runtime complexity of the proposed algorithm, including initialization, perturbations, packing, and cost computation, is $O(m^2)$, *i.e.*, satisfies the P* admissibility requirement of polynomial complexity.

The Matrix Floorplanner has been implemented in C++ and tested on standard floorplanning benchmarks, including MCNC and GSRC. When compared to previous work, the results show improvements across most performance metrics relative to similar algorithms, such as CBA-T. Additionally, the matrix floorplanner outperforms other algorithms that do not target thermal and TSV optimization in terms of area and wirelength

Three netlists obtained from industrial partners were also evaluated using the proposed

algorithm. The Matrix Floorplanner delivers high-quality results, with as little as 3.8% .

For future work, we propose developing a machine learning model to automate the selection of hyperparameters in optimization metaheuristics, such as simulated annealing, for floorplanning. This approach aims to alleviate the significant challenge of manually tuning hyperparameters, which are difficult to optimize due to their large number and dependency on the input netlist. By doing so, we hope to improve the quality of the final floorplan and reduce the time required for iterative testing by human experts.

Bibliography

[1] D. Al Saleh, Y. Safari, F. Amik, and B. Vaisband, "P* Admissible Thermal-Aware Matrix Floorplanner for 3D ICs," in *Proceedings of the IEEE 36th International System-on-Chip Conference (SOCC)*, pp. 1–6, 2023.

[2] Y. Safari, A. Corbier, D. Al Saleh, and B. Vaisband, "ARTSim: A Robust Thermal Simulator for Heterogeneous Integration Platforms," in *Proceedings of the IEEE 73rd Electronic Components and Technology Conference (ECTC)*, pp. 1187–1193, 2023.

[3] S. Chopra and S. Subramaniam, "A Review on Challenges for MOSFET Scaling," *International Journal of Innovative Science, Engineering & Technology*, vol. 2, no. 4, pp. 1055–1057, 2015.

[4] W. McMahon, A. Haggag, and K. Hess, "Reliability Scaling Issues for Nanoscale Devices," *IEEE Transactions on Nanotechnology*, vol. 2, no. 1, pp. 33–38, 2003.

[5] T.-C. Chen, "Overcoming research challenges for cmos scaling: Industry directions," in *2006 8th International Conference on Solid-State and Integrated Circuit Technology*

Proceedings, pp. 4–7, 2006.

[6] L. Eeckhout, "Heterogeneity in response to the power wall," *IEEE Micro*, vol. 35, pp. 2–3, July 2015.

[7] B. Vaisband and S. S. Iyer, "Global and Semi-Global Communication on Si-IF," in *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, Association for Computing Machinery, 2019.

[8] "Samsung Announces Availability of its Silicon-Proven 3D IC Technology for High-Performance Applications." Samsung Semiconductor Global, 2023.

[9] J. H. Lau, "Evolution, challenge, and outlook of TSV, 3D IC integration and 3d silicon integration," in *2011 International Symposium on Advanced Packaging Materials (APM)*, pp. 462–488, 2011.

[10] K. Dhananjay, P. Shukla, V. F. Pavlidis, A. Coskun, and E. Salman, "Monolithic 3D Integrated Circuits: Recent Trends and Future Prospects," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 3, pp. 837–843, 2021.

[11] A. Nahman, A. Fan, J. Chung, and R. Reif, "Wire-length Distribution of Three-Dimensional Integrated Circuits," in *Proceedings of the IEEE 1999 International Interconnect Technology Conference*, pp. 233–235, 1999.

[12] B. Vaisband, *3-D ICs as a Platform for Heterogeneous Systems Integration*. University of Rochester, 2017.

[13] V. F. Pavlidis *et al.*, *Three-Dimensional Integrated Circuit Design, Second Edition*. Morgan Kaufmann, 2017.

[14] X. Dong and Y. Xie, "System-Level Cost Analysis and Design Exploration for Three-Dimensional Integrated Circuits (3D ICs)," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 234–241, 2009.

[15] B. Vaisband and E. G. Friedman, "Analysis of Thermal Paths in 3-D Structures," in *Proceedings of the 37th Annual IEEE EDS/CAS Activities in Western New York Conference*, pp. 6–11, 2013.

[16] B. Vaisband, I. Savidis, and E. G. Friedman, "Thermal Conduction Path Analysis in 3-D ICs," in *Proceedings of the 2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 594–597, 2014.

[17] P. Zhou *et al.*, "3D-STAF: Scalable Temperature and Leakage Aware Floorplanning for Three-Dimensional Integrated Circuits," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (IC-CAD)*, pp. 590–597, November 2007.

[18] B. Shi and A. Srivastava, "Cooling of 3D-IC Using Non-Uniform Micro-Channels and Sensor Based Dynamic Thermal Management (Allerton)," in *Proceedings of the 49th*

Annual Allerton Conference on Communication, Control, and Computing, pp. 1400–1407, 2011.

[19] M. S. Bakir *et al.*, "3D Heterogeneous Integrated Systems: Liquid Cooling, Power Delivery, and Implementation," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 663–670, 2008.

[20] N. Arora, J. Hauser, and D. Roulston, "Electron and Hole Mobilities in Silicon as a Function of Concentration and Temperature," *IEEE Transactions on Electron Devices*, vol. 29, no. 2, pp. 292–295, 1982.

[21] J. Zhang *et al.*, "Thermal Stresses in 3D IC Inter-Wafer Interconnects," *Microelectronic Engineering*, vol. 82, no. 3, pp. 534–547, 2005.

[22] Y. Sun, L. Jiangbo, and G. Ding, "Modeling and Analysis of TSV Arrays with Different Ground and Signal Distributions in 2.5D/3D Integration Systems," *Journal of Physics: Conference Series*, vol. 1087, p. 052019, 2018.

[23] J. Cong, J. Wei, and Y. Zhang, "A Thermal-Driven Floorplanning Algorithm for 3D ICs," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (IC-CAD)*, pp. 306–313, 2004.

[24] H. Y. Zhu *et al.*, "Floorplanning for 3D-IC with Through-Silicon Via Co-Design Using Simulated Annealing," in *Proceedings of the IEEE Asia-Pacific Symposium on Electromagnetic Compatibility*, pp. 550–553, 2018.

[25] L. Xiao *et al.*, "Fixed-Outline Thermal-Aware 3D Floorplanning," in *Proceedings of the IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 561–567, 2010.

[26] F. Frantz *et al.*, "3D IC Floorplanning: Automating Optimization Settings and Exploring New Thermal-Aware Management Techniques," *Microelectronics Journal*, vol. 43, pp. 423–432, June 2012.

[27] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, "Chapter 10 - Metaheuristic Algorithms: A Comprehensive Review," in *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications* (A. K. Sangaiah, M. Sheng, and Z. Zhang, eds.), Intelligent Data-Centric Systems, pp. 185–231, Academic Press, 2018.

[28] C. Blum and A. Roli, "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison," *ACM Comput. Surv.*, vol. 35, p. 268–308, sep 2003.

[29] M. Z. Ali *et al.*, "A Novel Hybrid Cultural Algorithms Framework with Trajectory-Based Search for Global Numerical Optimization," *Information Sciences*, vol. 334-335, pp. 219–249, 2016.

[30] Y. Han, S. Roy, and K. Chakraborty, "Optimizing Simulated Annealing on GPU: A Case Study with IC Floorplanning," in *Proceedings of the 12th International Symposium on Quality Electronic Design (ISQED)*, pp. 1–7, 2011.

[31] B. Gwee and M. Lim, "A GA with heuristic-based decoder for IC floorplanning," *Integration*, vol. 28, no. 2, pp. 157–172, 1999.

[32] Q. Xu, S. Chen, and B. Li, "Combining the Ant system Algorithm and Simulated Annealing for 3D/2D Fixed-Outline Floorplanning," *Applied Soft Computing*, vol. 40, pp. 150–160, 2016.

[33] G. Chen, W. Guo, H. Cheng, X. Fen, and X. Fang, "VLSI Floorplanning Based on Particle Swarm Optimization," in *Proceedings of the 3rd International Conference on Intelligent System and Knowledge Engineering (ISKE)*, vol. 1, pp. 1020–1025, 2008.

[34] J.-M. Lin *et al.*, "TCG: A Transitive Closure Graph-Based Representation for Non-Slicing Floorplans," in *Proceedings of the IEEE/ACM Design Automation Conference (DAC)*, pp. 764–769, 2001.

[35] J.-M. Lin *et al.*, "TCG-S: Orthogonal Coupling of P*-Admissible Representations for General Floorplans," in *Proceedings of the IEEE/ACM Design Automation Conference (DAC)*, pp. 842–847, 2002.

[36] H. Murata *et al.*, "Rectangle-Packing Based Module Placement," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (IC-CAD)*, pp. 472–479, 1995.

[37] S.-Z. Zhang, Z.-Y. Zhao, C.-C. Feng, and L. Wang, "A Machine Learning Framework with Feature Selection for Floorplan Acceleration in IC Physical Design," *Journal of Computer Science and Technology*, vol. 35, pp. 468–474, 2020.

[38] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[39] K. Wang *et al.*, "Generative Adversarial Networks: Introduction and Outlook," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 4, pp. 588–598, 2017.

[40] A. Mirhoseini *et al.*, "A Graph Placement Methodology for Fast Chip Design," *Nature*, vol. 594, no. 7862, pp. 207–212, 2021.

[41] W. Huang *et al.*, "HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 5, pp. 501–513, 2006.

[42] X. Hao and F. Brewer, "Wirelength Optimization by Optimal Block Orientation," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (IC-CAD)*, pp. 64–70, 2005.

[43] B. Vaisband and E. G. Friedman, "Layer Ordering to Minimize TSVs in Heterogeneous 3-D ICs," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 1926–1929, 2016.

[44] "COMSOL Multiphysics."

[45] K. Bazargan, R. Kastner, and M. Sarrafzadeh, "3-D Floorplanning: Simulated Annealing and Greedy Placement Methods for Reconfigurable Computing Systems," in *Proceedings of the 10th IEEE International Workshop on Rapid System Prototyping. Shortening the Path from Specification to Prototype*, pp. 38–43, 1999.

[46] T. Ni *et al.*, "Temperature-Aware Floorplanning for Fixed-Outline 3D ICs," *IEEE Access*, vol. 7, pp. 139787–139794, 2019.

[47] S. Shi, X. Zhang, and R. Luo, "The thermal-aware floorplanning for 3d ics using carbon nanotube," in *2010 IEEE Asia Pacific Conference on Circuits and Systems*, pp. 1155–1158, 2010.

[48] "3-D IC Physical Design and 3-D Architecture Exploration."

[49] X. Li, Y. Ma, and X. Hong, "A Novel Thermal Optimization Flow Using Incremental Floorplanning for 3D ICs," in *Proceedings of the IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 347–352, 2009.

[50] Vidal-Obiols *et al.*, "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, 2021.

[51] J. Knechtel, E. F. Y. Young, and J. Lienig, "Planning massive interconnects in 3-d chips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 11, pp. 1808–1821, 2015.

Appendix A

Equations

$$\mathbf{c}_{\text{height}} = \sqrt{\frac{\mathbf{c}_{\text{area}}}{\text{AR}}} \tag{A.1}$$

$$\mathbf{c}_{\text{width}} = \frac{\mathbf{c}_{\text{area}}}{\mathbf{c}_{\text{height}}} \tag{A.2}$$

$$C = \frac{w_{\text{area}}}{n_{\text{area}}} \cdot A + \frac{w_{\text{wl}}}{n_{\text{wl}}} \cdot wl + \frac{w_{\text{temp}}}{n_{\text{temp}}} \cdot T_{\text{L}} + \frac{w_{\text{TSVs}}}{n_{\text{TSVs}}} \cdot N_{\text{TSVs}} \tag{A.3}$$

$$P_c = p_x \cdot r_x + p_y \cdot r_y + p_z \cdot r_z \tag{A.4}$$

$$T_{\text{L}} = \max\{P_c \cdot d_c \cdot \frac{N_{\text{TSVs}}^{n-1} + 1}{N_{\text{TSVs}}^{n} - 1}\} \tag{A.5}$$

$$L = \frac{Area - Area_{Ideal}}{Area_{Ideal}} + \frac{WL - WL_{Ideal}}{WLIdeal} + \frac{Temp - TempIdeal}{Temp_{Ideal}} \tag{A.6}$$

Appendix B

Tables

Paper	Constraint	Optimized metrics	Data Structure	Drawbacks
[23]	Peak temperature	Area, wire length, TSVs	CBA based on [35] limiting perturbations in 3D	Incomplete temperature estimation (one direction only)
[47]	Peak temperature	Area, TSVs	Tile-based floorplans (not P* admissible)	Relies on carbon nano-tubes to cool down circuit
[46]	Peak temperature	Area, wire length	B* Tree with no z-direction perturbations	No 3D perturbations, layer assignment overhead
[45]	N.A.	Area, wire length	TCG-S	Non-discreet z-coordinates
[40]	Area	Wirelength, congestion density	Vectors	No temperature reduction, Poor generalization

Table B.1: Table with various several state-of-art floorplanners, their cost functions, features, and drawbacks.

	CBA-T [23]	Li [49]	Matrix Floorplanner [1]
ami33	$4.14 \cdot 10^{-7}$	$3.61 \cdot 10^{-7}$	$3.97 \cdot 10^{-7}$
ami49	$1.84 \cdot 10^{-5}$	$1.24 \cdot 10^{-5}$	$1.38 \cdot 10^{-5}$
n100	$6.56 \cdot 10^{-8}$	$5.74 \cdot 10^{-8}$	$5.90 \cdot 10^{-8}$
n200	$6.91 \cdot 10^{-8}$	$6.83 \cdot 10^{-8}$	$5.61 \cdot 10^{-8}$
n300	$1.06 \cdot 10^{-7}$	$4.14 \cdot 10^{-7}$	$9.65 \cdot 10^{-8}$

Table B.2: Results of the evaluation of area (m^2) of the Matrix Floorplanner and comparison to previous work, for a 4-layer 3D IC.

	CBA-T [23]	Li [49]	Matrix Floorplanner [1]
ami33	0.024442	0.033111	0.024234
ami49	0.477646	0.198711	0.392008
n100	0.092456	0.072622	0.092008
n200	0.190886	0.179565	0.206000
n300	0.253837	0.257049	0.255564

Table B.3: Results of the evaluation of wirelength (m) of the Matrix Floorplanner and comparison to previous work, for a 4-layer 3D IC.

	CBA-T [23]	Li [49]	Matrix Floorplanner [1]
ami33	98	N/A	81
ami49	211	N/A	242
n100	1044	N/A	1207
n200	2021	N/A	1984
n300	2261	N/A	2144

Table B.4: Results of the evaluation of the number of TSVs of the Matrix Floorplanner and comparison to previous work, for a 4-layer 3D IC.

	CBA-T [23]	Li [49]	Matrix Floorplanner [1]
ami33	160	338	144
ami49	151	342	134
n100	158	273	130
n200	156	237	129
n300	167	270	113

Table B.5: Results of the evaluation of peak temperature (°C) of the Matrix Floorplanner and comparison to previous work, for a 4-layer 3D IC.

	CBA-T [23]	Li [49]	Matrix Floorplanner [1]
ami33	466	1.6	144
ami49	521	6.1	114
n100	4322	21	451
n200	6843	86	1035
n300	17484	256	1957

Table B.6: Results of the evaluation of runtime (s) of the Matrix Floorplanner and comparison to previous work, for a 4-layer 3D IC.

	Layers	Area w/o white space (m^2)	Area (m^2)	Percent of white space	WL (m)	TSVs	T (C)	Time (s)
10 blocks	2	$2.89 \cdot 10^{-5}$	$3.09 \cdot 10^{-5}$	6.9%	0.064178	11	76	15
30 blocks	2	$7.76 \cdot 10^{-5}$	$8.06 \cdot 10^{-5}$	3.8%	0.535507	56	94	168
30 blocks	4	$3.88 \cdot 10^{-5}$	$4.39 \cdot 10^{-5}$	13.1%	0.359770	95	133	68
50 blocks	4	$6.40 \cdot 10^{-5}$	$7.57 \cdot 10^{-5}$	18.3%	0.924614	206	135	67

Table B.7: Results of the evaluation of the proposed Matrix Floorplanner on netlists obtained from industrial partners.