Machine Learning for 3D IC Floorplanning

Dima Al Saleh

In this study design, we will first recall the research objectives, then we will elaborate on the plan to address the research question including the need for data collection, data sources, data preprocessing, and data analysis.

1. Study Objectives

We remind that the research question we seek to answer is the following "What is the performance of the best-known machine learning (ML) optimization models for three-dimensional (3D) integrated circuits (IC) floorplanning?". This question can be further subdivided as the following.

- Can an ML model be trained and used to perform floorplanning on 3D IC netlists?
- Do ML-based models perform better than search-based floorplanners on 3D ICs?
- Is there one overall best ML-based floorplanning model for all metrics (area, width, temperature, number of TSVs)?
- Should different models be used for different netlists, made by different companies?

By answering the above sub-questions, we seek to understand whether ML-based models should be used to floorplan 3D ICs. We also want to know which ML model is preferable for 3D floorplanning and the process behind training and using it. We recall that search-based floorplanning models often used in 2D floorplanning are slowly becoming obsolete. The increase in the number of blocks to include in the IC is exacerbating the runtime of these algorithms significantly. Moreover, adding layers vertically on top of one another increases the design space exponentially, rendering optimization more complex. In 3D floorplanning, temperature must also be optimized, since the circuit layers stacked on top of one another can cause an increase in temperature. Finally, through-silicon-vias (TSV), exclusively used in 3D ICs, also need to be optimized [1-4]. For these reasons, traditional algorithms are no longer efficient.

We plan to answer these questions by first, collecting the data necessary to train the ML models. This involves determining the data sources, the repositories, and the extraction process. Ensuring the quality of the collected data is also crucial; if the data has biases the model is likely to have biases as well [5]. Then we select ML optimization models popularly used for 2D ICs and adapt them to floorplan 3D ICs such as [6]. After training, the models will perform on the test dataset and according to the performance of the optimization on the following metrics area, wirelength, max temperature on the circuit, and number of TSVs, we would either recommend the use of these models or discourage it.

2. Data Collection

Similar to any ML algorithm, a dataset is needed to train the model on the task. Larger datasets have been shown to generally help improve the performance of models [7]. Therefore, a large dataset should be aimed for.

We remind you that ML floorplanning models are trained to determine the coordinates of blocks specified on a netlist [6]. A netlist is, therefore, a list of circuit components (blocks) with their specifications (width, length, connectivity, power). On the other hand, a floorplan is a netlist with the added x, y, and z coordinates for each block. To train a model to perform floorplanning on a netlist a dataset of netlists is required. Below, we detail how the data was collected.

2.1 Data Sources

The three main sources of IC netlists used for data collection are **GitHub** [8], **Southern Methodist University Platform** [9], and **University of Michigan** [10]. These sources contain IC netlists designed by a range of industry players, including Google, IBM, Xerox, and HP. These netlists are reliable and were also created to reflect the design characteristics and practical application needs within the industry [11]. Below we describe each data source.

GitHub [8] is an online platform for managing, sharing, and supporting source code, hosting over 100 million repositories. It includes a variety of search-based IC floorplanning projects which include netlists used to evaluate the performance of their floorplanner. Moreover, companies, including Google DeepMind [12], sometimes share intellectual property, including netlists and floorplans, on GitHub. However, depending on the repository, the quality of the netlists may vary. Some repositories contain lesser-known or experimental netlists, developed by researchers rather than industry players. This limitation and how it can be mitigated are detailed in Subsection 2.2.

Southern Methodist University (SMU) Platform [9] is an open-access online platform that hosts various netlists commonly used in floorplanning [11] [13][14], including those developed by the Microelectronics Center of North Carolina (MCNC) [15]. This resource is accessible to the public without any additional permissions.

University of Michigan (UM) Platform [10] is similarly an accessible online repository hosting several very-large-scale integration (VLSI) netlists frequently used in floorplanning tasks. This platform includes official IBM netlists and does not require special permissions, offering easy access for industry and academic users alike.

These sources were selected through the process of snowballing. Reviewing several seminal and state-of-the-art papers in the field of 2D and 3D IC floorplanning, a common list of data sources for netlists could be established [11-14]. Moreover, the SMU and MU platforms are well managed and properly maintained with easy access, which were essential criteria.

However, some limitations do arise from these data sources. On GitHub, anyone can upload floorplanning netlists; yet, these netlists may sometimes be overly simplistic or fail to reflect the characteristics of real ICs used in the industry. Details on handling these challenges are provided in Sub-subsection 2.2. Additionally, since netlists can be represented in various formats—such as multiple files versus a single file—and with differences in units for dimensions, and power, ensuring uniformity is essential. Therefore, data preprocessing steps must be adapted accordingly to achieve consistent representations across sources. The uniformization procedure will be automated through the use of specific scripts tailored to each format. Furthermore, duplicate netlists are likely to occur, necessitating mechanisms for duplicate removal. These mechanisms are discussed in Sub-subsection 2.2. Finally, due to the limited amount of data available, we suggest data expansion, which is often used in ML training [16]. Data expansion remains optional and can be used to

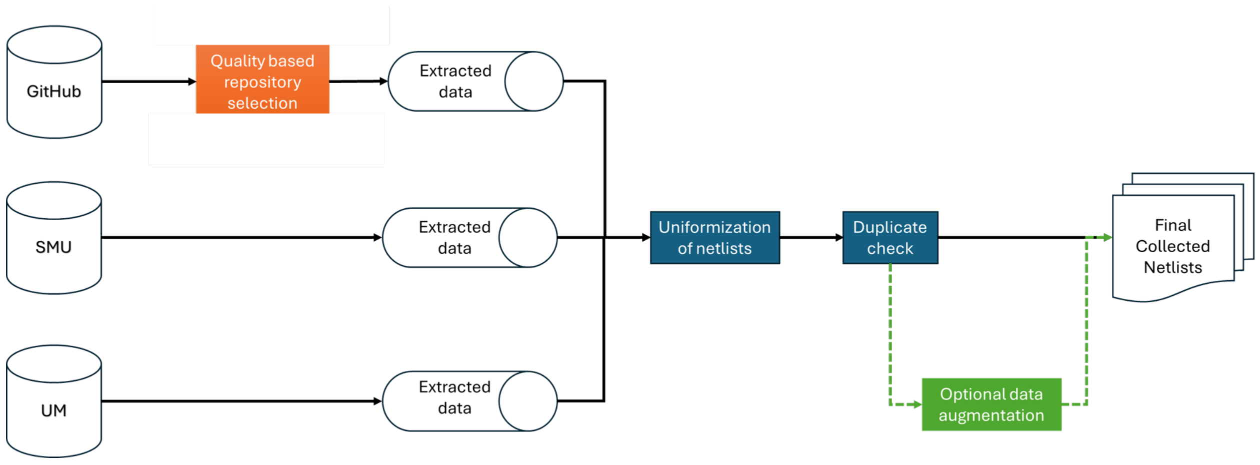

Figure 1: A schema showing the data collection process.

increase the model's accuracy. The methods used to implement data expansion are examined in Sub-subsection 2.4.

2.2 Data Repositories

Once the data sources were identified, we proceeded to select relevant repositories. As illustrated in Fig 1, repositories from SMU [9] and UM [10] did not require additional filtering. These platforms host high-quality netlists and provide comprehensive documentation on file structure and measurement units. Thus, the entirety of these platforms was included, as they enable training the floorplanning model with a diverse set of circuit netlists.

In contrast, GitHub required additional refinement. A query for "floorplanning IC netlist" returned over 26,400 results. Upon examining these results, it became evident that most did not contain meaningful netlists but rather general code files where "netlist" appeared. These repositories rarely included circuit netlists with industry significance. To address this, we employed a snowballing technique. Notably, the GitHub repository of Google's public domain netlists [17], referenced in [6] and [12], met our criteria and was retained due to its reputable circuit information. Using the same approach, we identified two additional repositories—Corblivar [18], used in [19-21], and HotSpot [22], referenced in [23].

To ensure thorough coverage, we conducted a targeted search using the query "ami33 n300 netlist." These netlists, which are frequently cited in floorplanning literature, served as anchors to locate potentially valuable repositories. This search returned 52 results, but after review, all contained netlists already gathered in prior stages.

In summary, repository selection involved a manual process of snowballing, supplemented by specific search queries to ensure completeness. The inclusion criteria focused on relevance and quality, targeting repositories with detailed documentation and significant netlists for circuit design. Selected GitHub repositories will be downloaded via scripts to reduce human error, while UM and SMU repositories will be downloaded manually due to the lack of API support.

2.3 Data Analysis

Two main data preprocessing steps are planned. First, we will standardize all netlists to ensure a consistent format and units (e.g., μm for width and height). Given that the data was sourced from multiple repositories, variations in netlist formats are expected. For each unique format, a dedicated script will be created, based on the provided documentation, to transform the netlists. The goal is to represent each netlist in the following files, which are useful for addressing the research questions.

- **Block specifications file** Each line in this file describes a new block in the circuit. The line begins with the block name (usually numbered), followed by a binary entry indicating if it's a hard or soft block. Hard blocks have fixed width and height, while soft blocks have a fixed area with flexible dimensions within a given aspect ratio. Depending on the block type, subsequent entries specify either width and height or area and aspect ratio. All dimensions are in micrometers. This file is essential for optimizing the circuit area.
- **Block connectivity file** This file lists connections between blocks, which is necessary for optimizing wirelength and the number of TSVs (since TSVs connect blocks with different z-coordinates).
- **Optional - Terminal connectivity file** This optional file specifies whether blocks are connected to a terminal or pin. While the Block Connectivity File shows inter-block connections, this file includes connections between blocks and specific pins with defined locations.
- **Optional - Power requirement file** This file is only needed if temperature optimization is performed. Each line contains a block name, followed by the power requirement of the block in watts (W). Depending on the power consumed by a block, power dissipation (heat dissipation) can be estimated, which reveals the temperature of part of the circuit.

The name, dimensions, connectivity, and power of each block are extracted directly from the netlist files downloaded from the various repositories. Scripts parse the raw data based on the documentation provided.

After standardizing the netlists, a script will check for and remove any duplicates. Duplicate netlists could skew the training process by disproportionately weighting certain circuits, which may lead to biased model outcomes. Removing these ensures that each unique netlist contributes equally to training.

2.4 Optional Data augmentation

To address the limited data available, we propose data augmentation to increase the variety of training samples. In this approach, some netlists with over 300 blocks can serve as templates for generating new netlists with fewer blocks. The potential number of netlists generated by selecting between 1 and 259 blocks from a set of 300 is theoretically:

$$\text{Total lists} = \sum_{k=1}^{259} \binom{300}{k}$$

where $\binom{300}{k}$ denotes the number of ways to choose k blocks out of 300. Only netlists with at least 10 blocks are retained for relevance. While power and dimensions remain consistent with the original netlist, connectivity between blocks must be updated to ensure no remaining block relies on connections to removed blocks.

Moreover, new netlists can be generated from scratch to mirror the original netlists properties, such as width and height distribution (average, minimum, maximum, and median width and height), block aspect ratios, and overall power density, within a certain margin. Given that netlists from different companies exhibit distinct characteristics, data expansion may be applied to each type individually. A model can then be trained and tested on each type of netlist separately. Data augmentation remains optional and can be bypassed if initial model testing indicates better performance without expanded data.

References

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

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

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

[4] 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.

[5] N. Mehrabi et al., "A Survey on Bias and Fairness in Machine Learning," ACM Computing Surveys, Vol. 54, no. 6, pp. 1–35, 2021.

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

[7] A. Bailly et al., "Effects of Dataset Size and Interactions on the Prediction Performance of Logistic Regression and Deep Learning Models", Computer Methods and Programs in Biomedicine, Vol. 213, 2022. [8] GitHub · *build and ship software on a single, Collaborative Platform.* GitHub. (n.d.). https://github.com/

[9] Index of / manikas/benchmarks. (n.d.). https://s2.smu.edu/ manikas/Benchmarks/

[10] ISPD06 floorplacement benchmarks. (n.d.). http://vlsicad.eecs.umich.edu/BK/ISPD06bench/

[11] S. Kai et al., "Tofu: A Two-Step Floorplan Refinement Framework for Whitespace Reduction". Design, Automation & amp; Test in Europe Conference &amp; Exhibition (DATE), pp. 1–5, 2023.

[12] A. G. Mirhoseini, A. S. Tsai, and P. S. Garg, "How AlphaChip Transformed Computer Chip Design", Google DeepMind, 2024.

[13] J. M. Lin et al., "A Fast thermal-Aware Fixed-Outline Floorplanning Methodology Based on Analytical Models," IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 1-8, 2018.

[14]W. Guan, et al., "A Novel Thermal-Aware Floorplanning and TSV Assignment With Game Theory for Fixed-Outline 3-D ICs," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 31, no. 11, pp. 1639-1652, Nov. 2023.

[15] https://www.mcnc.org/who-we-are/history/

[16] C. Shorten, T.M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning." J Big Data 6, 60, 2019.

[17] Google-Research. (n.d.). Google-research/circuit_training. GitHub. https://github.com/google-research/circuit_training

[18] DfX-NYUAD. (n.d.). DFX-Nyuad/Corblivar: Corblivar is a simulated-annealing-based floorplanning suite for 3D ICS. GitHub. https://github.com/DfX-NYUAD/Corblivar

[19] J. Knechtel, J. Lienig, and I. M. Elfadel, "Multi-Objective 3D Floorplanning with Integrated Voltage Assignment". ACM Trans. Des. Autom. Electron. Syst. 23, 2, Article 22, 27 pages, 2018.

[20] J. M. Lin et al. "Thermal-Aware Fixed-Outline Floorplanning Using Analytical Models With Thermal-Force Modulation," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 29, no. 5, pp. 985-997, May 2021.

[21] M. Amini et al., "Generalizable Floorplanner through Corner Block List Representation and Hypergraph Embedding." In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022.

[22] Uvahotspot. (n.d.). Uvahotspot/hotspot: Hotspot v7.0 is an accurate and fast thermal model suitable for use in architectural studies. GitHub. https://github.com/uvahotspot/HotSpot

[23] J. -H. Han, et al., "From 2.5D to 3D Chiplet Systems: Investigation of Thermal Implications with HotSpot 7.0," 21st IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (iTherm), pp. 1-6, 2022.