TP3D: TIMING-AWARE DIE-TO-DIE 3D MACRO PLACEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Macro placement is one of the most critical steps in chip design, exerting a significant impact on power, performance, and area (PPA), while existing works primarily focus on optimizing computationally efficient surrogate objectives, resulting in a marked gap in performance improvement. The emergence of 3D integrated circuits (3D-ICs) with stacked layers connected via hybrid bonding terminals further complicates accurate timing estimation at early stages of the design flow. To bridge the gap, we formulate timing-aware macro placement in 3D-IC design as a learning-based optimization problem, and propose **TP3D**, a novel learning-based, timing-aware 3D macro placement framework. Specifically, TP3D introduces a structured timing prediction model that decomposes intra-die interactions and latent bridge point based inter-die interactions. The predicted metrics are further integrated into the placement procedure in the form of timing masks to guide optimization. Extensive experiments demonstrate that TP3D outperforms state-of-the-art methods, achieving an average improvement of 10.8% in Worst Negative Slack (WNS) and 8.7% in Total Negative Slack (TNS), which are of great significance for enhancing chip performance. Notably, this work highlights the importance of integrating structured, learnable objective modeling for intractable objectives, compared to directly optimizing surrogate objectives. We release the code in the anonymous link: `https://anonymous.4open.science/r/TP3D-2B11`.

1 INTRODUCTION

As technology approaches fundamental physical limits, particularly beyond sub-3nm nodes, sustaining Moore's Law (Moore et al., 1965) faces challenges like escalating costs, heightened design complexity, and declining yields (Zhao et al., 2025a). In this context, 3D integrated circuits (3D-ICs) have emerged as a critical solution, enabling higher transistor density and extending integration along the z-axis. By vertically stacking multiple dies, 3D-ICs replace lengthy 2D interconnects with shorter vertical connections, significantly improving circuit performance through reduced wirelength, alleviated routing congestion, and enhanced timing (Shi et al., 2025a). Furthermore, advanced packaging facilitates the integration of heterogeneous chiplets for leading cost-effective performance (Murali & Lim, 2021).

Among 3D integration options, TSVs suffer from large pitch and parasitics (Zhao et al., 2025a; Dong et al., 2010), while M3D with MIVs faces cost, yield, and quality challenges (Zhao et al., 2025a; Samal et al., 2016; Zhu et al., 2021). This paper focuses on F2F bonded 3D-ICs (Jung et al., 2014; Song et al., 2015) with hybrid bonding terminals (HBTs), which provide dense vertical connectivity without the silicon area overhead of TSVs or MIVs (Ku et al., 2018).

Despite these benefits, 3D stacking substantially increases physical design complexity, especially for placement, where macro locations across tiers strongly affect routing and final PPA.

Placement can be categorized into macro placement and standard cell placement. As the initial and most critical step in the overall placement flow, macro placement exerts a profound impact on subsequent processes such as routing and further determines the final performance, power, and area (PPA) of the chip (MacMillen et al., 2000; Kim & Markov, 2012). Existing 3D macro placement methods are mainly pseudo-3D methods that rely on partitioning and 2D EDA tools (Shi et al., 2025a; Pentapati et al., 2020; Bamberg et al., 2020; Vanna-Iampikul et al., 2021), or true-3D analytical methods that directly optimize 3D placements (Murali et al., 2022; Liao et al., 2024; Kim et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2024; Zhao et al., 2025b). However, these methods still largely optimize surrogate metrics, and some true-3D formulations restrict cross-die macros from sharing the same horizontal position (Zhao et al., 2025b), reducing the searchable solution space. More importantly, HPWL and other surrogate objectives can have a marked gap with final timing metrics (Wang et al., 2024; Shi et al., 2025b), limiting their practical utility for timing-critical 3D macro placement.

To bridge the gap, we propose a novel learning-based, timing-aware 3D macro placement framework named **TP3D**. Unlike conventional methods that rely on surrogate metrics, TP3D incorporates predicted timing metrics into the macro placement stage to guide macro placement. Specifically, we design a **Two-Layer timing prediction model** that formulates the predicted timing metrics as a function of the product of relative distances between macros and their logical interactions. We develop two distinct relative distance models for intra-die and bridge-point-based inter-die macro interactions, respectively. Then we leverage graph neural networks (GNNs) to learn the ranking relationship between the predicted timing metrics and timing labels. Notably, inter-die macro interactions are not well captured by direct planar distances, since practical cross-die signal paths first travel horizontally toward an intermediate region and then vertically cross the dies. Therefore, for inter-die interactions, we introduce latent bridge points (LBP) as abstract modeling points in the timing prediction model. Our main contributions are summarized as follows:

- **A novel timing-aware framework for 3D macro placement**. We propose TP3D, a novel learning-based framework that brings downstream timing feedback into the early macro placement stage. TP3D predicts timing-oriented placement quality and converts the prediction into Two-Layer timing masks to guide timing-aware 3D macro placement.
- **Structured modeling of macro interactions**. We develop a Two-Layer timing prediction model that separately captures intra-die interactions and inter-die interactions. For inter-die interactions, we propose latent bridge points (LBP) that model inter-die interactions as two horizontal segments, enabling more reliable timing-based ranking of candidate 3D placements.
- **Zero-shot transferability**. TP3D exhibits zero-shot transferability to unseen designs. This property highlights its efficiency advantages, as it conserves significant training resources when tackling new designs.
- **SOTA experimental results with significant timing improvement**. Extensive experiments demonstrate that our method achieves the best timing metrics on most designs with 10.8% WNS improvement and 8.7% TNS improvement on average.

2 PRELIMINARY

2.1 3D MACRO PLACEMENT

A typical 3D-IC comprises a top die and a bottom die. Consider a netlist $(\boldsymbol{V}, \boldsymbol{E})$ where $\boldsymbol{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ denotes the set of macros and $\boldsymbol{E} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m\}$ denotes the connections between macros. For a net $\boldsymbol{e} \in \boldsymbol{E}$, its pins may be located either within a single die or across both the top and bottom dies. Single-die nets are modeled within their corresponding die, while cross-die nets are associated with vertical terminals that connect the two dies. Our task is to determine the positions (x, y, z) of all macros while satisfying physical constraints and optimizing the overall design quality.

2.2 METRICS

In EDA community, timing metrics are widely regarded as the primary criterion for assessing chip quality. However, obtaining the final timing results in early design stages requires running multiple downstream implementation steps (e.g., clock tree synthesis, routing, and so on), which substantially increases computational cost. Recent works have developed a range of efficiently computable surrogate metrics to approximate timing metrics. In the following, we introduce several key metrics to characterize placement quality, thereby clarifying the optimization objectives in the macro placement stage.

HPWL Half-perimeter wirelength (HPWL) is a surrogate metric during the placement stage and is commonly used as a surrogate metric of timing (Chen et al., 2006). For each net, HPWL denotes

the sum of the length and width of the bounding box. The total HPWL is the sum over all nets. For 3D architectures, prior work (Liao et al., 2024) further extends HPWL to account for cross-die connectivity. Although HPWL is efficient and widely adopted, recent works (Wang et al., 2024; Shi et al., 2025b) have shown a notable gap between HPWL and the final PPA. Accordingly, our work primarily focuses on timing, which directly reflects the chip performance.

WNS & TNS Worst negative slack (WNS) and total negative slack (TNS) are timing metrics reported by static timing analysis (STA) and are widely used to characterize timing quality and violations. WNS summarizes the most critical timing bottleneck in the design by reflecting the worst slack among all timing paths. TNS measures the overall severity of timing violations by aggregating the negative slacks across all violating paths. In 3D macro placement, macro positions and inter-die interaction paths directly affect interconnect delays, making WNS and TNS highly sensitive to placement results.

3 MOTIVATION

Existing 3D macro placement methods typically optimize timing through surrogate metrics such as HPWL. To test the reliability of this proxy, we collect placement-timing data and conduct a correlation analysis. Specifically, we generate placements using Analytical-M3D (Zhao et al., 2025b) and evaluate the resulting designs with the Open3DBench (Shi et al., 2025a) timing flow. By varying random seeds, we obtain diverse placements with different timing outcomes. We select 5 designs from Open3DBench and generate 200 placements per design, resulting in 1,000 samples in total. Figure 1 reports the Kendall correlation coefficients between HPWL, WNS, and TNS. The results show that the Kendall correlation coefficient between HPWL and WNS is only 0.004, and that between HPWL and TNS is even -0.331, indicating that HPWL is almost uninformative about final timing. This empirical gap motivates a timing-aware 3D macro placement framework instead of relying on HPWL optimization.



Figure 1: Kendall correlation coefficients.

These observations suggest a more direct path: learn the relationship from macro placement to timing explicitly, rather than treating timing as an indirect byproduct of surrogate optimization. In fact, we observe that the final timing results are jointly determined by macro locations and their logical interactions from an end-to-end perspective, so we model this dependency with a dedicated function, which we refer to as the Two-Layer timing prediction function in the subsequent sections.

4 METHOD

The overview of TP3D is shown in Figure 2. It has two main components. First, we develop a **Two-Layer timing prediction model**. We model intra-die macro interactions and bridge-point-based inter-die macro interactions separately and ultimately derive predicted metrics. Then we train the model with supervised learning. Second, we utilize the trained model to generate **Two-Layer timing masks**, which guide timing-aware macro placement. These components are elaborated in detail in Sections 4.1 and Sections 4.3, respectively.

4.1 TWO-LAYER TIMING PREDICTION MODEL

Based on the graph representation mentioned in Appendix B.1, we represent the netlist as a graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$. Then we leverage GNN to extract a d-dimensional node information vector $\boldsymbol{h}_v$ for each node v. All node information can be represented as a matrix $\boldsymbol{H} = (\boldsymbol{h}_1, \cdots, \boldsymbol{h}_{|V|})^\top \in \mathbb{R}^{|V| \times d}$, where $\boldsymbol{H} = \mathrm{GNN}_{\boldsymbol{\theta}}(\boldsymbol{G})$. We formulate the timing metrics $\hat{l}_\lambda$ as a function of two components: the positions $\boldsymbol{P} = (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in \mathbb{R}^{|V| \times 3}$ of the macros and the logical interactions $\boldsymbol{H}$. Following the supervised learning paradigm, we aim to fit the function f_λ such that $\hat{l}_\lambda = f_\lambda(\boldsymbol{H}, \boldsymbol{P})$. In fact, the timing metrics are determined by the horizontal relative positions between macros. Thus, they can be represented

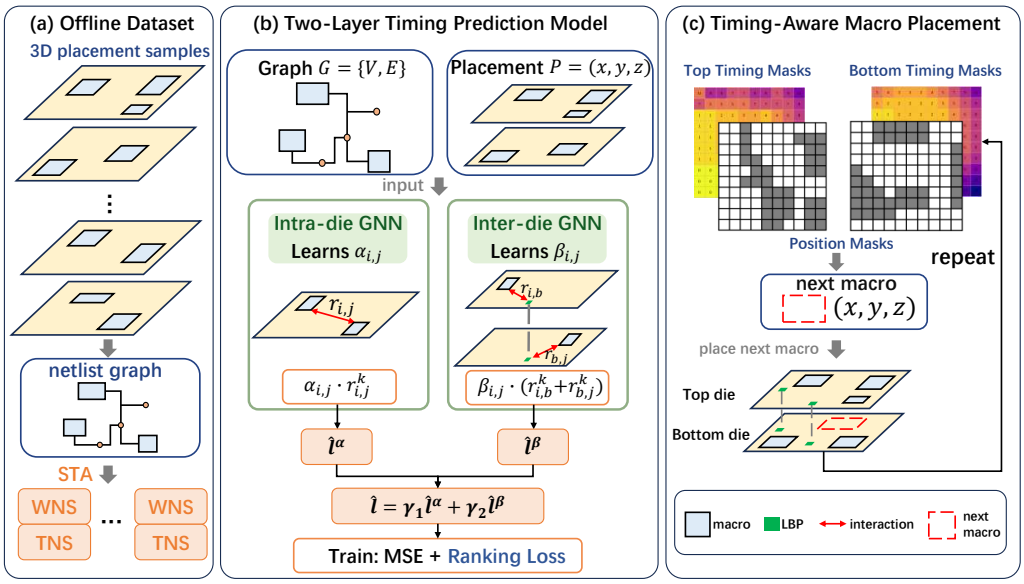


Figure 2: **Overview of TP3D.** (a) We construct the dataset for model training, which includes the macro placement results and timing labels. (b) We train the Two-Layer timing prediction model using the constructed offline dataset. "Two-Layer" refers to the fact that we establish two different models for inter-die and intra-die interactions, respectively. (c) We utilize the trained model to generate Two-Layer timing masks, which guide timing-aware macro placement.

by the horizontal relative distances $r_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ between macro i and macro j. Finally, the function can be further expressed as $\hat{l}_\lambda = f_\lambda(\boldsymbol{H}, \boldsymbol{R}, \boldsymbol{z})$, where $\{\boldsymbol{R}\}_{i,j} = r_{i,j}$.

It should be noted that both x and y here are continuous and vary over a wide range, whereas z can only take values of 0 or 1. $z = 0$ indicates the macro is on the bottom die; $z = 1$ on the top die. In fact, the interactions between macros on the same die differ from those between macros on different dies.

Figure 3 illustrates the difference with an example. In the left figure, macros on the same die are connected via planar wires, so their timing contribution can be modeled directly by a polynomial of their relative distance. In the right figure, macros on different dies follow a different interaction pattern: the path first moves horizontally toward an intermediate region and then crosses the dies vertically. We therefore employ two different strategies to model intra-die and inter-die interactions. Specifically, for macros on the same die, the function can be expressed as:

$$\hat{l}_{i,j}^{\alpha,(\lambda,k)} = \alpha_{i,j}^{(\lambda,k)}(\boldsymbol{H}) \cdot r_{i,j}^k \tag{1}$$

where K denotes a set of integers indicating the exponents in the polynomial, such as $\{1, 0, -1, -2\}$. $\alpha_{i,j}^{(\lambda,k)}(\boldsymbol{H})$ are distance coefficients, calculated as:

$$\alpha_{i,j}^{(\lambda,k)}(\boldsymbol{H}) = \frac{1}{2}\left(\boldsymbol{h}_i^\top \boldsymbol{M}^{(\lambda,k)} \boldsymbol{h}_j + \boldsymbol{h}_j^\top \boldsymbol{M}^{(\lambda,k)} \boldsymbol{h}_i\right) \tag{2}$$

where $\boldsymbol{M}^{(\lambda,k)}$ are learnable weight matrices.

For macros on different dies, we additionally introduce a latent bridge point (LBP) to model the inter-die interaction in the timing function.

LBP For macro i and j on different dies, with their respective horizontal coordinates (x_i, y_i) and (x_j, y_j), the latent bridge point $b_{i,j} = (x_{i,j}, y_{i,j})$ is parameterized as:

$$x_{i,j} = x_i + t_{i,j}^{(x)}(x_j - x_i) \tag{3}$$

$$y_{i,j} = y_i + t_{i,j}^{(y)}(y_j - y_i) \tag{4}$$

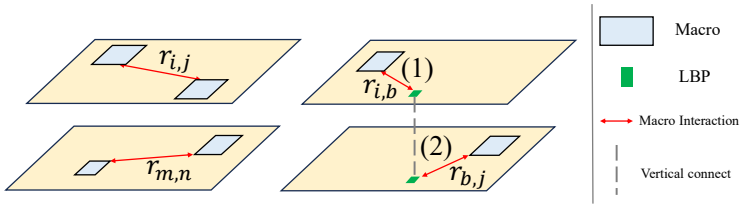


Figure 3: Intra-die and inter-die macro interaction modeling. Inter-die interactions are modeled through a latent bridge point (LBP).

where $t_{i,j}^{(x)}$ and $t_{i,j}^{(y)}$ are determined by the logical interactions between macro i and macro j:

$$t_{i,j}^{(n)} = \sigma\left(\frac{1}{2}\left(\boldsymbol{h}_i^\top \boldsymbol{N}_{bp}^{(n)} \boldsymbol{h}_j + \boldsymbol{h}_j^\top \boldsymbol{N}_{bp}^{(n)} \boldsymbol{h}_i\right)\right) \tag{5}$$

where $n \in [x, y]$, $\sigma(\cdot)$ denotes the sigmoid activation function, and $\boldsymbol{N}_{bp}^{(n)}$ are also learnable weight matrices. Therefore, $t_{i,j}^{(x)}$ and $t_{i,j}^{(y)}$ are mapped to $(0, 1)$, ensuring that the LBP lies within the axis-aligned region spanned by the two connected macros. Then, as shown in Figure 3, the interaction between macros on different dies is modeled as two horizontal segments from the macros to the LBP. So the function can be expressed as:

$$\hat{l}_{i,j}^{\beta,(\lambda,k)} = \beta_{i,j}^{(\lambda,k)}(\boldsymbol{H})(r_{i,b}^k + r_{j,b}^k) \tag{6}$$

where

$$r_{i,b} = \sqrt{(x_i - x_{i,j})^2 + (y_i - y_{i,j})^2}, \quad r_{j,b} = \sqrt{(x_j - x_{i,j})^2 + (y_j - y_{i,j})^2} \tag{7}$$

denote the distances from macros i and j to the LBP. In Equation 6, $\beta_{i,j}^{(\lambda,k)}(\boldsymbol{H})$ is a distance coefficient, calculated as:

$$\beta_{i,j}^{(\lambda,k)}(\boldsymbol{H}) = \frac{1}{2}\left(\boldsymbol{h}_i^\top \boldsymbol{N}^{(\lambda,k)} \boldsymbol{h}_j + \boldsymbol{h}_j^\top \boldsymbol{N}^{(\lambda,k)} \boldsymbol{h}_i\right) \tag{8}$$

where $\boldsymbol{N}^{(\lambda,k)}$ is a learnable weight matrix.

Two-Layer Timing Prediction Function Incorporating all macros on the same die and different dies, we obtain the Two-Layer timing prediction function:

$$\hat{l}_\lambda = \sum_{k \in K} \sum_{i,j} (\gamma_1 \hat{l}_{i,j}^{\alpha,(\lambda,k)} \cdot (1 - |z_i - z_j|) + \gamma_2 \hat{l}_{i,j}^{\beta,(\lambda,k)} \cdot |z_i - z_j|) \tag{9}$$

where γ_1, γ_2 are hyperparameters. By adjusting their values, the Two-Layer timing prediction function can be enabled to prioritize either the interactions between macros on the same die or those on different dies.

4.2 Model Training

To train the model, we build a dataset $\mathcal{D}$ with a set of D chip netlists, each modeled as a graph $\boldsymbol{G}_d$. For each netlist, we produce a set of M distinct placements $\{\boldsymbol{P}_{d,m}\}_{m=1}^M$ by Analytical-M3D (Zhao et al., 2025b) with varied seeds. Later procedures—such as CTS, routing, and STA are run with Open3DBench (Shi et al., 2025a) to generate the timing labels $l_{d,m}^{(\lambda)}$ for each $\lambda \in \{\text{WNS}, \text{TNS}\}$. Finally, we acquire the dataset

$$\mathcal{D} = \left\{\boldsymbol{G}_d, \boldsymbol{P}_{d,m}, l_{d,m}^{(\lambda)}\right\} \tag{10}$$

Since timing labels are produced by complex downstream EDA stages, directly regressing their absolute values with MSE alone is difficult and sensitive to noise. We therefore combine MSE with a

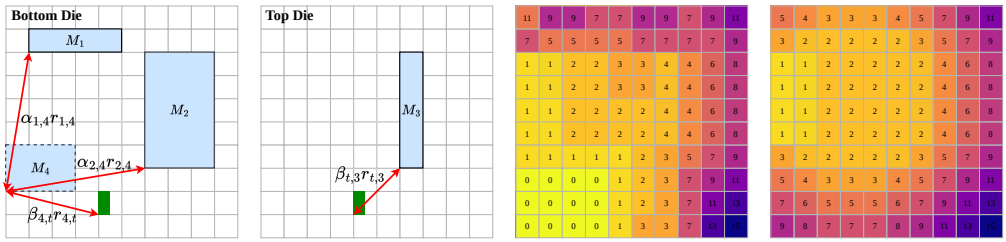


Figure 4: **An example of placement with Two-Layer timing masks.** M_1 and M_2 are placed in bottom die, while M_3 is placed in top die. M_4 is to be placed. The lower figure is Two-Layer timing masks of M_4. Furthermore, we first filter candidate cells with position masks and then select the legal position corresponding to the minimum timing value as the position of M_4.

robust pairwise ranking loss, which encourages the predicted relative order of placements to match the timing labels. The ranking loss is given by:

$$\mathcal{L}_{\text{Rank}} = \frac{1}{DM(DM-1)} \sum_{\lambda} \sum_{i=1}^{DM} \sum_{j \neq i} w_{i,j}^{(\lambda)} \cdot \mathcal{L}_{i,j}^{(\lambda)} \tag{11}$$

where the pairwise term $\mathcal{L}_{i,j}^{(\lambda)}$ and weight $w_{i,j}^{(\lambda)}$ are detailed in Appendix B.7. Finally, we adopt the MSE loss and the ranking loss to train the model:

$$\mathcal{L} = \delta_1 \cdot \mathcal{L}_{\text{MSE}} + \delta_2 \cdot \mathcal{L}_{\text{Rank}} \tag{12}$$

where δ_1,δ_2 are hyperparameters.

4.3 Macro Placement With Two-Layer Timing Masks

After training, we have obtained the Two-Layer timing prediction model. Yet, it remains a question of how to apply this model to macro placement. In the macro placement stage, the chip canvas is typically discretized into $N \times N$ grids. It should be noted that the Two-Layer timing prediction function is actually a function of the distances between macros and their logical interactions. Drawing on the idea of generating wiremasks in MaskPlace (Lai et al., 2022), we can also use the Two-Layer timing prediction function to generate a similar mask, which we term Two-Layer timing masks. Since the next macro can be placed on either the top die or the bottom die, Two-Layer timing masks consist of two $N \times N$ masks, the bottom timing mask and the top timing mask. To guarantee placement legality, we also maintain position masks for the top and bottom dies, where legal grid cells satisfy boundary and non-overlap constraints with already placed macros. During candidate selection, the timing masks are stacked with the binary position masks to obtain combined masks, and the final location is selected from the minimum value of the combined masks. Compared with wiremask, we only replace the physical distances between macros in the calculation process with weighted abstract distances. The generation detail of Two-Layer timing masks is illustrated in Figure 4. Taking M_4 as an example, the bottom timing mask of M_4 is calculated as follows:

$$T_{bottom} = \sum_{k,\lambda} (\gamma_1 \alpha_{1,4}^{(\lambda,k)} r_{1,4}^k + \gamma_1 \alpha_{2,4}^{(\lambda,k)} r_{2,4}^k + \gamma_2 \beta_{4,3}^{(\lambda,k)} (r_{4,b}^k + r_{3,b}^k)) \tag{13}$$

where $r_{1,4}, r_{2,4}, r_{4,b}, r_{3,b} \in \mathbb{R}^{N \times N}$, which denote the candidate-grid distances. In particular, $r_{4,b}$ and $r_{3,b}$ are the LBP-based two-segment distances between the candidate position of M_4 and the placed macro M_3 on the opposite die. The calculation of T_{top} is similar to that of T_{bottom}. We then stack the corresponding position masks with the timing masks and determine the position of M_4 according to the minimum value in the combined masks. By repeating the above process, we can place all macros and obtain a complete macro placement. In addition, Wiremask-EA (Shi et al., 2023) integrates the evolutionary algorithm (EA) (Back, 1996) into wiremask and generates more high-quality macro placement results. Drawing inspiration from this work, we replace wiremask in Wiremask-EA with Two-Layer timing masks and further optimize the placement results using EA. Through this approach, we obtain more macro placements with promising timing results. The full implementation algorithm is provided in Appendix B.3.

5 EXPERIMENT

5.1 EXPERIMENTAL SETUP

Benchmarks We conduct our experiments on 7 designs of Open3DBench (Shi et al., 2025a), with detailed design statistics presented in Table 2 in Appendix B.4. This work is implemented based on OpenROAD (Ajayi et al., 2019), and the workflow supports accurate PPA evaluation, including 3D timing and power analysis. We utilize the derived 3D timing results to construct the dataset and conduct corresponding evaluations.

Baselines We compare our method with several recent advanced methods. FlexPlanner (Zhong et al., 2024) is a multi-modal reinforcement learning (RL) method, which is widely used in the 3D floorplan stage. Due to the similarity between the floorplan and macro placement stages, this method can be well applied to macro placement. We remove two metrics used in the floorplan stage, retaining HPWL for macro placement. Open3DBench (Shi et al., 2025a) proposes two 3D macro placement methods, which are representative methods in pseudo-3D methods. **(1)** Open3D-Tiling is based on the skyline packing algorithm (Wei et al., 2011) for 3D macro placement. **(2)** Open3D-DMP is a projection-based method that leverages 2D macro placement tools DREAMPlace (Lin et al., 2019; Chen et al., 2023) for 3D macro placement. Analytical-M3D (Zhao et al., 2025b) is the current advanced analytical method, based on 3D electrostatic field structures and macro rotation.

Dataset Collection and Training We use *ariane133, ariane136, bp_be, bp_fe*, and *bp_multi* for model training, while *bp_quad* and *swerv_wrapper* are excluded from the training set to verify the generalization ability of our model. For each training design, we generate 200 distinct placements with Analytical-M3D (Zhao et al., 2025b) by varying random seeds, resulting in 1000 placements in total. These placements are then fed into Open3DBench (Shi et al., 2025a) for 3D timing analysis, yielding a dataset that contains both macro placement results and timing labels. We use 80% of this dataset for training and the remaining 20% for validation. More details of the model architecture are provided in Appendix B.6. All experiments are conducted on a single machine equipped with NVIDIA GeForce GTX 3090 GPUs and Intel(R) Xeon(R) E5-2667 v4 CPUs (3.20GHz).

5.2 MAIN RESULTS

After obtaining 3D macro placement results using different methods, we run Analytical-M3D (Zhao et al., 2025b) for standard cell placement, and run Open3DBench (Shi et al., 2025a) to obtain the final timing results. As shown in Table 1, in terms of WNS, TP3D consistently outperforms four other baselines across all designs, achieving an average improvement of 10.8%. For TNS, TP3D outperforms in most designs, with an average improvement of 8.7%. Notably, on the unseen design *swerv_wrapper*, TP3D improves WNS by up to 26% and TNS by up to 41%. In addition, we present the curves of the training process in Figure 10 in Appendix C.2. The figure shows that the Kendall correlation coefficient between predicted WNS (pWNS) and WNS labels exceeds 0.6 and that between predicted TNS (pTNS) and TNS labels exceeds 0.7. This indicates that our model can effectively capture the complex mapping between macro placement results and the final timing metrics, and the predicted values are accurate in the sense of ranking. Notably, the Kendall correlation coefficients between HPWL and WNS, TNS are only 0.004 and -0.331, respectively. This well explains why using our predicted timing metrics to guide macro placement yields better timing results than other HPWL-based methods. We also report the runtime of all methods in Table 5 in Appendix C.1.

5.3 CORRELATION ANALYSIS

Figure 5 presents the Kendall correlation coefficients between the predicted solutions and labels. Specifically, the Two-Layer timing prediction model yields three predicted solutions: pWNS, pTNS, and Pred. The metric "Pred" denotes the average of pWNS and pTNS. Pred is employed in Section 4.3 to guide macro placement. Notably, the Kendall correlation coefficients between pWNS and WNS labels, as well as between pTNS and TNS labels, reach 0.606 and 0.704, respectively. This indicates that TP3D can effectively learn the complex mapping between macro placement results and timing metrics. The Kendall correlation coefficients between Pred and WNS, and between Pred and TNS,

Table 1: **TNS** (×ns) **and WNS** (×ns) **for macro placement derived by different approaches.** For TNS and WNS, higher is better. The results for FlexPlanner, Open3D-DMP, Analytical-M3D and TP3D are obtained from three independent runs with different random seeds. And we report the mean and standard deviations (mean±std) of each metric. We mark the best results in **bold**

		ariane133	ariane136	bp_be	bp_fe	bp_multi	bp_quad	swerv_wrapper
FlexPlanner	WNS	-0.17 (±0.04)	-1.14 (±0.03)	-0.39 (±0.03)	-0.96 (±0.08)	-13.95 (±0.16)	-1.38 (±0.12)	-0.66 (±0.16)
	TNS	-124.29 (±55.03)	-2903.32 (±150.81)	-34.41 (±3.37)	-111.11 (±7.68)	-5976.74 (±3347.22)	-14232.16 (±1875.01)	-317.56 (±130.15)
Open3D-DMP	WNS	-0.16 (±0.02)	-1.20 (±0.02)	-0.36 (±0.02)	-0.91 (±0.02)	-5.74 (±0.42)	-1.03 (±0.08)	-0.61 (±0.21)
	TNS	-83.91 (±21.25)	-3180.00 (±140.44)	-32.83 (±2.91)	-95.17 (±20.22)	-3657.42 (±754.87)	-11532.00 (±495.34)	-291.14 (±160.48)
Open3D-Tiling	WNS	-0.23	-1.19	-0.64	-0.92	-6.58	-1.46	-1.08
	TNS	-161.67	-3256.48	-59.02	-137.50	-3273.00	-17645.16	-704.94
Analytical-M3D	WNS	-0.15 (±0.03)	-1.14 (±0.04)	-0.34 (±0.01)	-0.87 (±0.03)	-5.45 (±0.01)	-0.74 (±0.09)	-0.53 (±0.12)
	TNS	-66.10 (±15.58)	-2963.30 (±85.93)	-30.31 (±0.56)	-101.67 (±3.29)	**-2514.33 (±204.48)**	**-4097.09 (±425.85)**	-228.85 (±53.47)
TP3D	WNS	**-0.13 (±0.01)**	**-1.07 (±0.01)**	**-0.32 (±0.01)**	**-0.85 (±0.01)**	**-5.37 (±0.01)**	**-0.59 (±0.03)**	**-0.39 (±0.03)**
	TNS	**-52.02 (±6.14)**	**-2602.25 (±34.66)**	**-27.69 (±0.65)**	**-84.83 (±1.73)**	-2837.15 (±51.56)	-5250.15 (±216.33)	**-134.75 (±29.44)**



Figure 5: Kendall correlation coefficients between predicted metrics and true metrics. The prefix "p" denotes the predicted values.

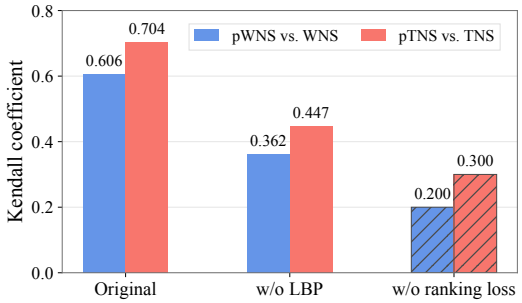
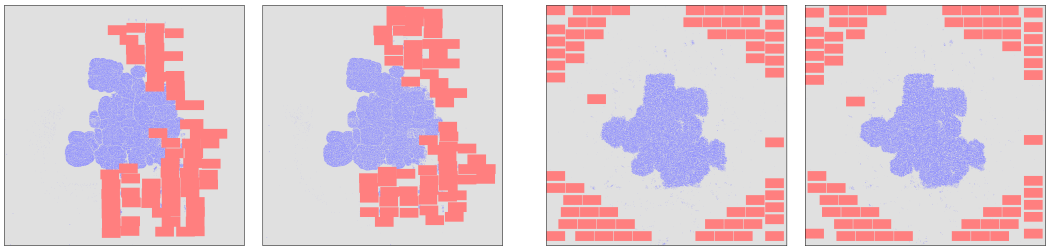


Figure 6: Ablation study results for LBP modeling and ranking loss. Hatched bars show the upper bounds of the unstable ranges.

are 0.567 and 0.534, respectively. This well explains why the placement results obtained by TP3D outperform those of other works in terms of timing metrics.

5.4 Ablation Studies

We conduct ablation studies to isolate the effects of the latent bridge point (LBP) modeling and the ranking loss in the Two-Layer timing prediction model. The results are summarized in Figure 6. In the w/o LBP variant, inter-die interactions are degraded to the same distance modeling as intra-die interactions. This variant noticeably decreases the Kendall correlation coefficients for both WNS and TNS, demonstrating that LBP-based inter-die modeling is critical for capturing the unique timing behavior of 3D-ICs. The MSE-only variant without ranking loss fails to converge stably, validating the necessity of the ranking loss for learning reliable relative placement quality. Since WNS and TNS are produced after multiple downstream EDA stages, directly regressing their absolute values is difficult and sensitive to noise. In contrast, ranking loss better matches our placement objective, where the model only needs to identify which candidate placement is better from a timing perspective.



(a) Analytical-M3D: HPWL = 4.45 $\times 10^5$um , WNS = -0.16 ns, TNS = -87.92 ns

(b) TP3D: HPWL = 7.55 $\times 10^5$um, WNS = -0.13 ns, TNS = -45.88 ns

Figure 7: **Visualization of placement results on *ariane133*.** The left and right images of each subfigure represent the bottom and top dies, respectively. Macros are marked in red, while standard cells are in blue. The evaluation metrics are labeled under each figure.

5.5 Analysis

Case Study We visualize the placement results of the SOTA method Analytical-M3D (Zhao et al., 2025b) and TP3D in Figure 7. In this case, Analytical-M3D exhibits excellent HPWL results, but TP3D outperforms it on the timing metrics. Analytical-M3D mainly completes macro placement by optimizing HPWL; therefore, its placement results tend to have macros clustered in the center of the canvas, as this can achieve better HPWL performance. However, such placement results are likely to cause routing difficulties, thereby degrading the timing metrics. TP3D tends to place macros at the edges of the bottom and top dies, respectively, while reserving the center area for standard cells. This is a behavior of experienced designers, because although this strategy may increase HPWL, it can improve routing and optimize the final timing results (Chiou et al., 2016). We are quite surprised that TP3D can discover this strategy without any prior knowledge. In addition, for macros on the same die, TP3D selectively aggregates and disperses macros. We believe this is an important strategy of TP3D in improving timing metrics: macros with critical interactions are placed as close as possible, while those with insignificant interactions are placed as far as possible, thereby achieving an optimal balance between pursuing low interaction latency and improving routing. For macros on different dies, TP3D tends to reduce their bridge-point-based interaction distances. Such behavior helps reduce inter-die interaction latency and ultimately achieves superior timing performance. Through these different interaction tendencies for macros on the same die and different dies, TP3D finally achieves better timing results. We also provide the visualized results of TP3D on all designs in Appendix C.6.

Comparison Studies We further study the sensitivity of TP3D to the polynomial exponents K, GNN depth, and training data scale. By default, we use $K = \{1, 0, -1, -2\}$ in Equations 1, 6 and 9; additional choices of K are reported in Table 7 in Appendix C.4. Results with 3, 5, and 7 GNN layers are shown in Figure 11 in Appendix C.5. We also train TP3D with 1k, 2k, and 3k placement samples in Appendix C.3, showing that the performance is already close to saturation with 1k samples.

6 Conclusion and Limitations

In this work, we propose TP3D, a novel learning-based framework for 3D macro placement. TP3D predicts timing metrics using the Two-Layer timing prediction model and leverages the predicted timing to further guide macro placement. Experimental results demonstrate that TP3D outperforms existing SOTA methods in timing metrics, which is of great significance for enhancing chip performance. Our work also has some limitations. We only consider timing metrics in the current study, while thermal metrics, another key PPA metric, are not included. Future work can incorporate thermal metrics for multi-objective optimization.

AI Use Statement

In this work, we used generative AI tools for language polishing and LaTeX formatting assistance only. We have not used generative AI tools for research ideation, method design, or the generation of experimental results, and tasks such as data collection and human-subject studies are not applicable to this work. All research ideas, algorithms, code, and experiments were conceived, implemented,

and validated by the authors. We have reviewed all AI-assisted text. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Ethics Statement

This work targets electronic design automation (EDA) for 3D integrated circuits and is evaluated exclusively on public, open-source circuit benchmarks (Open3DBench). It involves no human subjects, no private or sensitive data, and no dual-use application beyond the general purpose of improving semiconductor design automation. We do not identify direct negative societal impacts of this work, and no conflicts of interest or sponsorship issues apply.

Reproducibility Statement

We make substantial efforts to ensure reproducibility. All experiments are run on the public Open3DBench benchmarks with the open-source OpenROAD flow. Design statistics are given in Table 2, the hyperparameter settings in Table 3, and the model architecture in Table 4. The graph representation, the wiremask background, and the Two-Layer timing mask computation together with the complete placement procedure are fully specified in Appendix B.1, Appendix B.2, and Appendix B.3 (Algorithms 1 and 2), and the ranking loss is detailed in Appendix B.7. Anonymous source code is available at `https://anonymous.4open.science/r/TP3D-2B11`.

REFERENCES

T Ajayi, D Blaauw, TB Chan, CK Cheng, VA Chhabria, DK Choo, M Coltella, S Dobre, R Dreslinski, M Fogaça, et al. OpenROAD: Toward a self-driving, open-source digital layout implementation tool chain. In *Proceedings of Government Microcircuit Applications and Critical Technology Conference*, pp. 1105–1110, Albuquerque, NM, 2019.

Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

Lennart Bamberg, Alberto García Ortiz, Lingjun Zhu, Sai Pentapati, Da Eun Shim, and Sung Kyu Lim. Macro-3d: A physical design methodology for face-to-face-stacked heterogeneous 3d ics. In *DATE*, pp. 37–42, 2020.

Kyungwook Chang, Saurabh Sinha, Brian Cline, Raney Southerland, Michael Doherty, Greg Yeric, and Sung Kyu Lim. Cascade2d: A design-aware partitioning approach to monolithic 3d ic with 2d commercial tools. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8. IEEE, 2016.

Tung-chieh Chen, Zhe-wei Jiang, Tien-chang Hsu, Hsin-chen Chen, and Yao-wen Chang. A high-quality mixed-size analytical placer considering preplaced blocks and density constraints. In *2006 IEEE/ACM International Conference on Computer Aided Design*, pp. 187–192, 2006. doi: 10.1109/ICCAD.2006.320084.

Yifan Chen, Zaiwen Wen, Yun Liang, and Yibo Lin. Stronger mixed-size placement backbone considering second-order information. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9, San Francisco, CA, 2023.

Chien-Hsiung Chiou, Chin-Hao Chang, Szu-To Chen, and Yao-Wen Chang. Circular-contour-based obstacle-aware macro placement. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 172–177. IEEE, 2016.

Xiangyu Dong, Jishen Zhao, and Yuan Xie. Fabrication cost analysis and cost-aware design space exploration for 3-d ics. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(12):1959–1972, 2010.

Zijie Geng, Jie Wang, Ziyan Liu, Siyuan Xu, Zhentao Tang, Mingxuan Yuan, Jianye Hao, Yongdong Zhang, and Feng Wu. Reinforcement learning within tree search for fast macro placement. In *Forty-first International Conference on Machine Learning*, 2024.

Zijie Geng, Jie Wang, Ziyan Liu, Siyuan Xu, Zhentao Tang, Shixiong Kai, Mingxuan Yuan, Jianye Hao, and Feng Wu. Lamplace: Learning to optimize cross-stage metrics in macro placement. In *The Thirteenth International Conference on Learning Representations*, 2025.

Meng-Kai Hsu, Valeriy Balabanov, and Yao-Wen Chang. Tsv-aware analytical placement for 3-d ic designs based on a novel weighted-average wirelength model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(4):497–509, 2013.

Moongon Jung, Taigon Song, Yang Wan, Yarui Peng, and Sung Kyu Lim. On enhancing power benefits in 3d ics: Block folding and bonding styles perspective. In *Proceedings of the 51st Annual Design Automation Conference*, pp. 1–6, 2014.

Andrew B Kahng, Lutong Wang, and Bangqi Xu. Tritonroute: The open-source detailed router. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(3):547–559, 2020.

Donggyu Kim, Minjae Kim, Junseok Hur, Jakang Lee, Jinoh Cho, and Seokhyeong Kang. TA3D: Timing-aware 3D IC partitioning and placement by optimizing the critical path. In *Proceedings of the ACM/IEEE International Symposium on Machine Learning for CAD*, pp. 1–7, New York, NY, 2024.

Jinwoo Kim, Lingjun Zhu, Hakki Mert Torun, Madhavan Swaminathan, and Sung Kyu Lim. Micro-bumping, hybrid bonding, or monolithic? a ppa study for heterogeneous 3d ic options. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 1189–1194. IEEE, 2021.

Myung-Chul Kim and Igor L Markov. Complx: A competitive primal-dual lagrange optimization for global placement. In *Proceedings of the 49th Annual Design Automation Conference*, pp. 747–752, 2012.

Bon Woong Ku, Kyungwook Chang, and Sung Kyu Lim. Compact-2d: A physical design methodology to build commercial-quality face-to-face-bonded 3d ics. In *Proceedings of the 2018 International Symposium on Physical Design*, pp. 90–97, 2018.

Yao Lai, Yao Mu, and Ping Luo. Maskplace: Fast chip placement via reinforced visual representation learning. *Advances in Neural Information Processing Systems*, 35:24019–24030, 2022.

Yao Lai, Jinxin Liu, Zhentao Tang, Bin Wang, Jianye Hao, and Ping Luo. Chipformer: Transferable chip placement via offline decision transformer. In *International Conference on Machine Learning*, pp. 18346–18364. PMLR, 2023.

Peiyu Liao, Yuxuan Zhao, Dawei Guo, Yibo Lin, and Bei Yu. Analytical die-to-die 3-D placement with bistratal wirelength model and GPU acceleration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 43(6):1624–1637, 2024.

Yibo Lin, Shounak Dhar, Wuxi Li, Haoxing Ren, Brucek Khailany, and David Z Pan. DREAMPlace: Deep learning toolkit-enabled GPU acceleration for modern VLSI placement. In *Proceedings of ACM/IEEE Design Automation Conference*, pp. 1–6, Las Vegas, NV, 2019.

Jingwei Lu, Hao Zhuang, Ilgweon Kang, Pengwen Chen, and Chung-Kuan Cheng. eplace-3d: Electrostatics based placement for 3d-ics. In *Proceedings of the 2016 on International Symposium on Physical Design*, pp. 11–18, 2016.

Don MacMillen, Raul Camposano, Dwight Hill, and Thomas W Williams. An industrial view of electronic design automation. *IEEE transactions on computer-aided design of integrated circuits and systems*, 19(12):1428–1448, 2000.

Gordon E Moore et al. Cramming more components onto integrated circuits, 1965.

Gauthaman Murali and Sung Kyu Lim. Heterogeneous 3d ics: Current status and future directions for physical design technologies. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 146–151. IEEE, 2021.

Gauthaman Murali, Sandra Maria Shaji, Anthony Agnesina, Guojie Luo, and Sung Kyu Lim. Art-3D: Analytical 3D placement with reinforced parameter tuning for monolithic 3D ICs. In *Proceedings of the International Symposium on Physical Design*, pp. 97–104, Virtual, Canada, 2022.

Min Pan and Chris Chu. Fastroute: A step to integrate global routing into placement. In *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, pp. 464–471, 2006.

Shreepad Panth, Kambiz Samadi, Yang Du, and Sung Kyu Lim. Shrunk-2-D: A physical design methodology to build commercial-quality monolithic 3-D ICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(10):1716–1724, 2017.

Sai Surya Kiran Pentapati, Kyungwook Chang, Vassilios Gerousis, Rwik Sengupta, and Sung Kyu Lim. Pin-3d: A physical synthesis and post-layout optimization flow for heterogeneous monolithic 3d ics. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pp. 1–9, 2020.

Sandeep Kumar Samal, Deepak Nayak, Motoi Ichihashi, Srinivasa Banna, and Sung Kyu Lim. Monolithic 3d ic vs. tsv-based 3d ic in 14nm finfet technology. In *2016 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*, pp. 1–2. IEEE, 2016.

Yunqi Shi, Ke Xue, Song Lei, and Chao Qian. Macro placement by wire-mask-guided black-box optimization. *Advances in Neural Information Processing Systems*, 36:6825–6843, 2023.

Yunqi Shi, Chengrui Gao, Wanqi Ren, Siyuan Xu, Ke Xue, Mingxuan Yuan, Chao Qian, and Zhi-Hua Zhou. Open3dbench: Open-source benchmark for 3d-ic backend implementation and ppa evaluation, 2025a. URL https://arxiv.org/abs/2503.12946.

Yunqi Shi, Siyuan Xu, Shixiong Kai, Xi Lin, Ke Xue, Mingxuan Yuan, and Chao Qian. Timing-driven global placement by efficient critical path extraction. In *2025 Design, Automation & Test in Europe Conference (DATE)*, pp. 1–7. IEEE, 2025b.

Taigon Song, Arthur Nieuwoudt, Yun Seop Yu, and Sung Kyu Lim. Coupling capacitance in face-to-face (f2f) bonded 3d ics: Trends and implications. In *2015 IEEE 65th Electronic Components and Technology Conference (ECTC)*, pp. 529–536. IEEE, 2015.

Pruek Vanna-Iampikul, Chengjia Shao, Yi-Chen Lu, Sai Pentapati, and Sung Kyu Lim. Snap-3d: A constrained placement-driven physical design methodology for face-to-face-bonded 3d ics. In *Proceedings of the 2021 International Symposium on Physical Design*, pp. 39–46, 2021.

Zhihai Wang, Zijie Geng, Zhaojie Tu, Jie Wang, Yuxi Qian, Zhexuan Xu, Ziyan Liu, Siyuan Xu, Zhentao Tang, Shixiong Kai, et al. Benchmarking end-to-end performance of AI-based chip placement algorithms. *arXiv preprint*, 2024.

Lijun Wei, Wee-Chong Oon, Wenbin Zhu, and Andrew Lim. A skyline heuristic for the 2D rectangular packing and strip packing problems. *European Journal of Operational Research*, 215(2):337–346, 2011.

Clifford Wolf, Johann Glaser, and Johannes Kepler. Yosys-a free verilog synthesis suite. In *Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip)*, volume 97, 2013.

Yunxuan Zhao, Lancheng Zou, and Bei Yu. Physical design for advanced 3D ICs: Challenges and solutions. In *Proceedings of the International Symposium on Physical Design*, Austin, TX, 2025a. In press.

Yuxuan Zhao, Peiyu Liao, Siting Liu, Jiaxi Jiang, Yibo Lin, and Bei Yu. Analytical heterogeneous die-to-die 3-D placement with macros. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 44(2):402–415, 2025b.

Ruizhe Zhong, Xingbo Du, Shixiong Kai, Zhentao Tang, Siyuan Xu, Jianye Hao, Mingxuan Yuan, and Junchi Yan. Flexplanner: Flexible 3d floorplanning via deep reinforcement learning in hybrid action space with multi-modality representation. *Advances in Neural Information Processing Systems*, 37:49252–49278, 2024.

Lingjun Zhu, Lennart Bamberg, Sai Surya Kiran Pentapati, Kyungwook Chang, Francky Catthoor, Dragomir Milojevic, Manu Komalan, Brian Cline, Saurabh Sinha, Xiaoqing Xu, et al. High-performance logic-on-memory monolithic 3-d ic designs for arm cortex-a processors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29(6):1152–1163, 2021.

A Related Work

In this section, we discuss the related work. Existing methods on 3D placement can be categorized into pseudo-3D methods and true-3D methods.

Pseudo-3D Methods Pseudo-3D (Chang et al., 2016; Panth et al., 2017; Ku et al., 2018; Bamberg et al., 2020; Pentapati et al., 2020; Vanna-Iampikul et al., 2021; Shi et al., 2025a) methods decouple the partitioning and placement stages and adopt 2D placement tools to determine instance positions. Cascade2D (Chang et al., 2016) realizes a design flow of placement following partitioning. Shrunk2D (Panth et al., 2017) halves the macro area by reducing the macro size, thus enabling the global placement of all components on a single chip. Compact2D (Ku et al., 2018) improves on Shrunk2D by calibrating RC parasitics and optimizing the introduction of compact 3D integration through post-planning layer partitioning. Pin-3D (Pentapati et al., 2020) emphasizes the pin projection concept that enables direct component integration on a 2D plane, and further achieves 3D placement through partitioning. Snap-3D (Vanna-Iampikul et al., 2021) extends support for cell placement across two chips by halving the cell height and leveraging 2D tools to place all cells on a single chip. Macro-3D (Bamberg et al., 2020) adopts a logic-memory architecture, allowing 2D routing tools to place HBTs in a via-like manner. Open3DBench (Shi et al., 2025a) proposes a 3D flow for evaluation, which is implemented based on several open-source tools (Pan & Chu, 2006; Wolf et al., 2013; Ajayi et al., 2019; Lin et al., 2019; Kahng et al., 2020; Kim et al., 2021) and presents two methods for macro placement. Although the aforementioned methods yield legal 3D placements, these methods fail to fully explore the entire solution space, and their performance is particularly sensitive to the results of partitioning.

True-3D Methods True-3D (Hsu et al., 2013; Lu et al., 2016; Murali et al., 2022; Liao et al., 2024; Kim et al., 2024; Zhao et al., 2025b) methods generally adopt analytical approaches, which perform cell positioning within 3D cuboidal regions based on smooth wirelength and density models. Representative works include NTUPlace3-3D (Hsu et al., 2013), which employs a bell-shaped density model to account for the inserted through-silicon vias (TSVs), and ePlace-3D (Lu et al., 2016), which models density constraints as a 3D electrostatic field. Recently, Analytical-3D (Liao et al., 2024) uses a two-layer wirelength model to enhance the accurate approximation of 3D wirelength. Based on this model, Analytical-M3D (Zhao et al., 2025b) implemented a preprocessing operator to support mixed-size 3D placement of macros and standard cells. Art-3D (Murali et al., 2022) integrates an academic true 3D placement engine with tool-based commercial PPA evaluation. Another recent work, TA-3D (Kim et al., 2024), proposes a framework that enables timing-aware partitioning and true 3D placement. Although the aforementioned methods have achieved significant advances in true 3D placement, these methods rely heavily on estimations or surrogate metrics, which still exhibit a marked gap with the final timing metrics.

B Implementation Details

B.1 Graph Representation

We represent the netlist as a hypergraph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$ on the macro-level to avoid explicitly modeling the large-scale standard cells as nodes (Geng et al., 2025). Each vertex $\boldsymbol{v} \in \boldsymbol{V}$ corresponds to a macro, and each edge $\boldsymbol{e} \in \boldsymbol{E}$ denotes the interactions between macros. To account for the influence of standard cells without expanding the graph, we associate each edge with feature vectors that summarize macro-to-macro interactions mediated by standard cells. This compact graph representation can capture the influence of standard cells without incurring significant computational overhead. A typical circuit netlist contains hundreds of macros together with a large number of standard cells; as shown in Figure 8, we quantify the structural relationship between two macros by counting net-connected paths between them: a path consisting of k cells between two macros is referred to as a k-path, and k is termed the path depth.

Since each net consists of both input and output components, we should take their directions into consideration. Therefore, we distinguish between directed and undirected k-paths. For each node pair (i, j), let $\mathcal{N}_{ij}^{(k)}$ be the number of directed k-paths from i to j, and $\mathcal{N}_{ij}^{\prime(k)}$ the number of undirected k-paths connecting them. We insert an edge (i, j) if there exists a directed path of depth at most D or

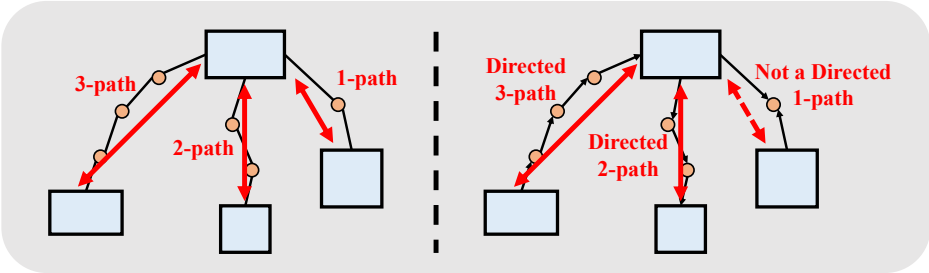


Figure 8: **Visualization of directed and undirected k-paths.** Blue blocks denote macros, and orange dots denote cells. In the left figure, we do not consider the directions. In the right figure, we consider both directed and undirected paths.

an undirected path of depth at most D'. Here, D and D' specify the maximum depths considered for directed and undirected paths, respectively.

To capture the "information closeness" between macros, the EDA community often adopts the dataflow metric. In our formulation, dataflow is incorporated as an edge attribute to summarize cell-level connectivity. Specifically, for directed paths, we define

$$f_{i,j}^{(D)} = \sum_{0 \le k \le D} \left(\frac{1}{2}\right)^k \cdot \mathcal{N}_{ij}^{(k)}, \tag{14}$$

and for undirected paths, we analogously define

$$f_{i,j}^{\prime(D')} = \sum_{0 \le k \le D'} \left(\frac{1}{2}\right)^k \cdot \mathcal{N}_{ij}^{\prime(k)}. \tag{15}$$

Formally, a netlist is modeled as a Graph $\mathcal{G}$ with node features H and edge features E. Each node feature vector contains five channels:

$$\mathbf{h}_i = (\text{size_x}_i, \text{size_y}_i, \text{node_area}_i, \text{sqrt_node_node}_i, \text{num_pins}_i). \tag{16}$$

For an edge connecting nodes i and j, we construct an edge feature of dimension $d_e = D + D' + 4$:

$$\mathbf{e}_{ij} = (f_{ij}^{(D)}, \mathcal{N}_{ij}^{(0)}, \cdots, \mathcal{N}_{ij}^{(D)}, f_{ij}^{\prime(D')}, \mathcal{N}_{ij}^{\prime(0)}, \cdots, \mathcal{N}_{ij}^{\prime(D')}). \tag{17}$$

In this work, we set $D = 9$ and $D' = 1$.

B.2 Masks

Due to the discrete and constrained nature of macro placement, prior 2D work (Lai et al., 2022; Shi et al., 2023; Lai et al., 2023) commonly adopts mask-based heuristics to guide candidate selection at low cost. Two typical examples are: (i) position mask, a binary mask that identifies locations where a macro can be placed without causing overlapping; and (ii) wiremask, a cost mask that estimates how much HPWL would increase if the next macro is placed at any grid cell. The concept of wiremask was first introduced by (Lai et al., 2022). Figure 9 visualizes wiremask through a simple 2D macro placement example. Suppose M_1 and M_2 have been fixed on the canvas, and we are about to insert M_3. In the left subfigure, two nets are highlighted by their axis-aligned bounding boxes, shown in red and green. The current masked HPWL is obtained by summing the half-perimeters of these boxes, i.e., HPWL $= w_1 + h_1 + w_2 + h_2$. Given a candidate placement for M_3, the resulting change in HPWL can be evaluated deterministically. In the left subfigure, the green grid cells indicate feasible locations where M_3 can be placed without causing overlaps. The right subfigure shows the wiremask as a 2D cost map, where each entry corresponds to the additional HPWL incurred when assigning M_3 to that grid cell. Since this increment is an explicit function of macro coordinates, the wiremask can be generated efficiently with minimal overhead. With the wiremask available, the placement decision for the next macro reduces to selecting the one that minimizes the induced HPWL increase among feasible grid cells.

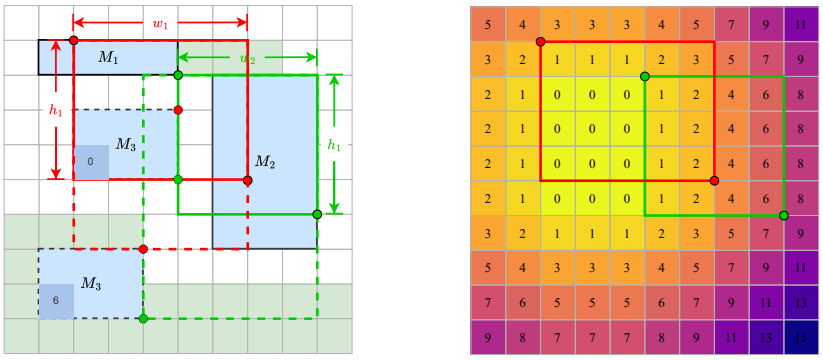


Figure 9: An example from (Geng et al., 2024) to illustrate the concept of the wiremask.

B.3 Two-Layer Timing Masks & Placement

Two-Layer timing masks share a similar form with wiremask, as both reflect a process of numerical increase during macro placement. However, wiremask captures the increase in HPWL, whereas Two-Layer Timing Masks capture the increase in timing. These masks help us achieve macro placements with better timing results. The detailed calculation process of the Two-Layer Timing Masks is elaborated in Algorithm 1. We then leverage the generated timing masks to guide macro placement. Drawing on the design philosophy of Wiremask-EA (Shi et al., 2023), we incorporate the evolutionary algorithm (EA) into Two-Layer timing masks. Specifically, we place the macros one by one by stacking the timing masks with the binary position masks to obtain combined masks, and then selecting the positions corresponding to the minimum values in the combined masks, thereby yielding an initial placement solution. In the subsequent evolutionary process, mutation is implemented by randomly swapping the positions of two macros, which transforms a single placement solution into a set of candidate placement solutions. We then select the optimal solution from these candidates as the final placement result. More algorithm details are demonstrated in Algorithm 2.

Algorithm 1 Calculation of Two-Layer Timing Masks

Input: Placed macros P, macro to be placed v_i, $\alpha_{i,j}^{(\lambda,k)}$, $\beta_{i,j}^{(\lambda,k)}$, $t_{i,j}^{(x)}$, $t_{i,j}^{(y)}$
Parameters: Number of canvas grid n, sum weight η_λ
Output: Two-Layer timing masks T_{top} and T_{bottom}
Initialize T_{top}, T_{bottom} as $n \times n$ grids of 0
for v_j **in** P **do**
  Get the grid position of v_j, (x_j, y_j, z_j)
  Initialize G_x, G_y as $n \times n$ grids
  for $x \leftarrow 0$ to $n-1$ **do**
    for $y \leftarrow 0$ to $n-1$ **do**
      $G_x[x][y] \leftarrow x$
      $G_y[x][y] \leftarrow y$
  $R \leftarrow \sqrt{(G_x - x_j)^2 + (G_y - y_j)^2}$, quantifying the distance from grid cells to (x_j, y_j)
  $B_x \leftarrow G_x + t_{i,j}^{(x)}(x_j - G_x)$, $B_y \leftarrow G_y + t_{i,j}^{(y)}(y_j - G_y)$
  $R_{i,b} \leftarrow \sqrt{(G_x - B_x)^2 + (G_y - B_y)^2}$, $R_{j,b} \leftarrow \sqrt{(x_j - B_x)^2 + (y_j - B_y)^2}$
  if $z_j = 0$ **then**
    $T_{bottom} \leftarrow T_{bottom} + \sum_\lambda \eta_\lambda \sum_k \alpha_{i,j}^{(\lambda,k)} \cdot R^k$
    $T_{top} \leftarrow T_{top} + \sum_\lambda \eta_\lambda \sum_k \beta_{i,j}^{(\lambda,k)} \cdot (R_{i,b}^k + R_{j,b}^k)$
  else
    $T_{top} \leftarrow T_{top} + \sum_\lambda \eta_\lambda \sum_k \alpha_{i,j}^{(\lambda,k)} \cdot R^k$
    $T_{bottom} \leftarrow T_{bottom} + \sum_\lambda \eta_\lambda \sum_k \beta_{i,j}^{(\lambda,k)} \cdot (R_{i,b}^k + R_{j,b}^k)$
Return: Two-Layer timing masks T_{top} and T_{bottom}

Algorithm 2 Placement Algorithm

Input: Netlist $\mathcal{G} = (\mathcal{V}_m, \mathcal{E}_n)$, $\alpha_{i,j}^{(\lambda,k)}$,$\beta_{i,j}^{(\lambda,k)}$
Parameters: Number of initial random turns N_{init}, Number of evolutionary turns N_{ea}, masking value $\mathcal{M}$ larger than feasible timing scores
Output: Placement P

For each macro $v_i \in \mathcal{V}_m$, compute its interaction weight as $\frac{1}{2}\sum_{j,k}\left|\alpha_{i,j}^{(\lambda,k)} + \beta_{i,j}^{(\lambda,k)}\right| A_i$, where A_i is the area of v_i
Order all macros in decreasing order based on their interaction weight, obtaining the ranked macro $\{v_1^*, \cdots, v_n^*\}$
for n **in** N_{init} **do**
  Random initialize position for each v_i^*, denoted by p_i
  Initialize placed-macros position as empty set P
  Initialize best placement P^* with best value V^*
  for each v_i^* **do**
    Generate position masks P_{top} and P_{bottom} from the boundary and non-overlap constraints
    Calculate Two-Layer timing masks T_{top} and T_{bottom} based on placed-macros P, $\alpha_{i,j}^{(\lambda,k)}$ and $\beta_{i,j}^{(\lambda,k)}$ with the process in Algorithm 1
    Construct combined masks $S_{top} = T_{top} \odot P_{top} + (1 - P_{top}) \odot \mathcal{M}$ and $S_{bottom} = T_{bottom} \odot P_{bottom} + (1 - P_{bottom}) \odot \mathcal{M}$
    if $\min S_{top} \leq \min S_{bottom}$ **then**
      $Q \leftarrow (\arg\min S_{top},\ 1)$
    else
      $Q \leftarrow (\arg\min S_{bottom},\ 0)$
    Select q from Q which is closest to the macro initial position p_i
    Incorporate v_i^* with position q to P
  Generate the predicted value V of final placement P
  if $V < V^*$ **then**
    $V^* \leftarrow V$
    $P^* \leftarrow P$
for n **in** N_{ea} **do**
  Initialize position for each v_i^* as p_i^* from P^*
  Swap positions of two randomly selected macro
  Initialize placed-macros position as empty set P
  for each v_i^* **do**
    Generate position masks P_{top} and P_{bottom} from the boundary and non-overlap constraints
    Calculate Two-Layer timing masks T_{top} and T_{bottom} based on placed-macros P, $\alpha_{i,j}^{(\lambda,k)}$ and $\beta_{i,j}^{(\lambda,k)}$ with the process in Algorithm 1
    Construct combined masks $S_{top} = T_{top} \odot P_{top} + (1 - P_{top}) \odot \mathcal{M}$ and $S_{bottom} = T_{bottom} \odot P_{bottom} + (1 - P_{bottom}) \odot \mathcal{M}$
    if $\min S_{top} \leq \min S_{bottom}$ **then**
      $Q \leftarrow (\arg\min S_{top},\ 1)$
    else
      $Q \leftarrow (\arg\min S_{bottom},\ 0)$
    Select q from Q which is closest to the macro initial position p_i^*
    Incorporate v_i^* with position q to P
  Generate the predicted value V of final placement P
  if $V < V^*$ **then**
    $V^* \leftarrow V$
    $P^* \leftarrow P$
Return: The best placement P^*

B.4 BENCHMARKS

In Table 2, we provide the statistics of all designs from benchmarks. In this work, we use the same design settings in all experiments.

Designs	**# Macros**	**# Nets**	**# Cells**	**# Pins**
ariane133	132	204K	168K	643k
ariane136	136	209K	171K	658k
bp_be	10	66K	51K	193k
bp_fe	11	43K	33K	121k
bp_multi	26	179K	152K	509k
bp_quad	220	1448K	1135K	4243k
swerv_wrapper	28	117K	98K	378k

Table 2: Detailed statistics of all designs.

B.5 HYPERPARAMETERS

In addition, we provide the detailed values of all hyperparameters used in the experiments in Table 3. $\gamma_1,\ \gamma_2$ denote the weights of intra-die and inter-die interactions in the Two-Layer timing prediction function. $\delta_1,\ \delta_2$ denote the weights between the MSE loss and the ranking loss. lr and batch_size denote the learning rate and batch size adopted during the model training, respectively. N denotes the grid size used in the placement process. $\eta_1,\ \eta_2$ denote the weights for the proportions of pWNS and pTNS in the placement process. C denotes the truncation value in the ranking loss function. $D,\ D'$ represent the depth of direct and indirect paths in the graph representation, respectively. $N_{init},\ N_{ea}$ denote the numbers of initial and evolutionary turns used in placement in Algorithm 2.

Configuration	Value
$\gamma_1,\ \gamma_2$	0.5, 0.5
$\delta_1,\ \delta_2$	0.001, 1
lr	0.001
batch_size	60
the number of grid N	100
$\eta_1,\ \eta_2$	0.5, 0.5
clip C	30
$D,\ D'$	9, 1
$N_{init},\ N_{ea}$	50, 20

Table 3: Hyperparameters used in the experiments

B.6 MODEL ARCHITECTURE

In this section, we provide a detailed description of the architecture and certain parameters of the Two-Layer timing prediction model. The model primarily consists of two identical GNNs: one is dedicated to extracting intra-die interactions, while the other captures inter-die interactions. After

training, the two GNNs develop distinct feature preferences. Both GNNs share the same graph representation input mentioned in Appendix B.1. The structure of the model is presented in Table 4.

module	layer name	layer size	output size
edge encoder×2	MLP_1	16×256 `ReLU()` 256×256	(num of edges, 256)
node encoder×2	MLP_2	3×256 `ReLU()` 256×256	(num of nodes, 256)
graph encoder×2	GNN $\times 5$	256×256 for all MLPs	(num of nodes, 256)
output encoder×2	MLP_3	256×256 `ReLU()` 256×256	(num of nodes, 256)
timing decoder×2	Linear Layer $\times 2 \times 4$	256×256 for all layers	(num of nodes, num of nodes, 16)

Table 4: Hyperparameters of the prediction model

B.7 Ranking Loss

In this work, we specifically employ the pairwise ranking loss, which aims to optimize the correlation between predicted metrics and labels. We expect the relative ranking order among the predicted values to remain consistent with that among the labels, as this enables the model to directly assess the relative quality of new layouts. Previous studies (Chen et al., 2023) have proposed a ranking loss function; however, it carries the risk of infinite loss values during training, which ultimately leads to training failure. To address this issue, we propose a more robust ranking loss function. Specifically, for predicted values $\hat{l_i}$ and their corresponding labels l_i, the ranking loss is defined as follows:

$$\mathcal{L}_{\text{Rank}} = \frac{1}{DM(DM-1)} \sum_{\lambda} \sum_{i=1}^{N} \sum_{j \neq i} w_{i,j}^{(\lambda)} \cdot \mathcal{L}_{i,j}^{(\lambda)} \tag{18}$$

where $\mathcal{L}_{i,j}^{(\lambda)}$ is piecewise pairwise loss. It is given by:

$$\mathcal{L}_{i,j}^{(\lambda)} = \begin{cases} \log\left(1 + \exp\left(-p_{i,j}^{(\lambda)}\right)\right), & p_{i,j}^{(\lambda)} \geq 0 \\ \log\left(1 + \exp\left(p_{i,j}^{(\lambda)}\right)\right) - p_{i,j}^{(\lambda)}, & p_{i,j}^{(\lambda)} < 0 \end{cases} \tag{19}$$

where

$$p_{i,j}^{(\lambda)} = \text{sign}(l_i^{(\lambda)} - l_j^{(\lambda)}) \cdot \text{clip}(\hat{l}_i^{(\lambda)} - \hat{l}_j^{(\lambda)}, -C, C) \tag{20}$$

where clip C is a constant. In Equation 18, $w_{i,j}^{(\lambda)}$ is a pairwise importance weight. It is given by:

$$w_{i,j}^{(\lambda)} = \left| \frac{\exp(l_{i,\text{norm}}^{(\lambda)}) - \exp(l_{j,\text{norm}}^{(\lambda)})}{\sum_k \exp(l_{k,\text{norm}}^{(\lambda)}) + \epsilon} \right| \tag{21}$$

where $l_{i,\text{norm}}^{(\lambda)}$, $l_{j,\text{norm}}^{(\lambda)}$ and $l_{k,\text{norm}}^{(\lambda)}$ denote the values after normalization and ϵ is a small value.

C More Results

C.1 Runtime Analysis

We report the runtime of TP3D and all baseline methods in Table 5. We report the time in the macro placement stage for FlexPlanner, Open3D-DMP, Open3D-Tiling, and TP3D. Since Analytical-M3D is a mixed-size method that places both macros and standard cells simultaneously, we report the time of the global placement stage.

	ariane133	ariane136	bp_be	bp_fe	bp_multi	bp_quad	swerv_wrapper
FlexPlanner	275	266	14	11	37	687	49
Open3D-DMP	161	122	92	160	138	578	162
Open3D-Tiling	54	52	20	18	46	332	34
Analytical-M3D	235	231	192	99	175	2287	148
TP3D	291	272	5	5	15	768	17

Table 5: Runtime (s) of compared methods to obtain the macro placement results.

C.2 Training Details

We present curves of key metrics during the training process in Figure 10. It can be observed that as the number of training epochs increases, the Kendall coefficients between pWNS and WNS, as well as between pTNS and TNS, continuously rise, ultimately exceeding 0.6 and 0.7, respectively. This clearly demonstrates the effectiveness of the Two-Layer timing prediction model.

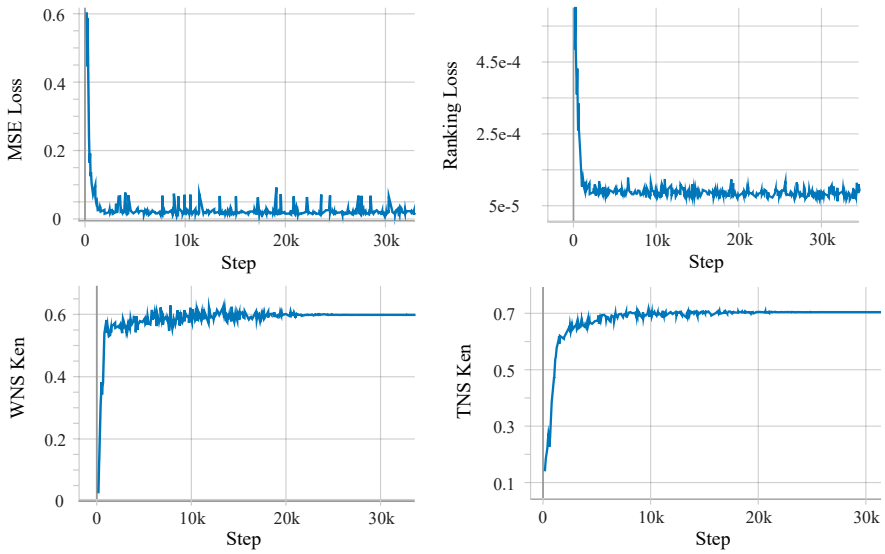


Figure 10: Training curves of four key metrics, including MSE loss, Ranking loss, Kendall coefficients between pWNS and WNS, and Kendall coefficients between pTNS and TNS.

C.3 Comparison Study on Training Data Scale

To evaluate the sensitivity of TP3D to the number of training samples, we train the model with 1k, 2k, and 3k placement samples while keeping other settings unchanged. The results are summarized in Table 6. As the amount of training data increases, TP3D shows slight improvements on several

designs, while the overall performance is already close to saturation with 1k samples. This indicates that TP3D is data-efficient and remains effective under limited labeled placement data.

		ariane133	ariane136	bp_be	bp_fe	bp_multi	bp_quad	swerv_wrapper
1k	WNS	-0.13	-1.07	-0.32	-0.85	-5.37	-0.59	-0.39
	TNS	-52.02	-2602.25	-27.69	-84.83	-2837.15	-5250.15	-134.75
2k	WNS	-0.12	-1.05	-0.32	-0.88	-5.33	-0.53	-0.38
	TNS	-51.43	-2588.67	-28.32	-83.36	-2875.02	-5133.28	-146.18
3k	WNS	-0.12	-1.01	-0.32	-0.83	-5.31	-0.53	-0.39
	TNS	-51.08	-2536.11	-26.42	-81.19	-2807.46	-4987.65	-143.22

Table 6: Comparison study on different training data scales. TNS and WNS are reported in ns, and higher values are better.

C.4 Comparison Study on K

K is mainly used as the exponent in the Two-Layer timing prediction function mentioned in Section 4.1. To explore the impact of K values on the Two-Layer timing prediction model, we conduct comparative experiments by changing the value of K while keeping other frameworks and parameters unchanged. The final experimental results are shown in Table 7.

K		ariane133	ariane136	bp_be	bp_fe	bp_multi	bp_quad	swerv_wrapper
{1,0,-1}	WNS	-0.17 (±0.01)	-1.12 (±0.01)	-0.28 (±0.01)	-0.91 (±0.02)	-5.39 (±0.01)	-0.78 (±0.05)	-0.46 (±0.03)
	TNS	-115.185 (±15.94)	-2834.75 (±79.19)	-25.30 (±0.99)	-93.46 (±5.64)	-2949.75 (±795.05)	-7958.65 (±1060.39)	-163.29 (±24.02)
{1,0,-2}	WNS	-0.16 (±0.01)	-1.13 (±0.01)	-0.29 (±0.02)	-0.95 (±0.01)	-5.41 (±0.02)	-0.99 (±0.10)	-0.41 (±0.14)
	TNS	-96.79 (±22.14)	-2900.98 (±34.69)	-24.97 (±1.49)	-108.97 (±8.76)	-2302.1 (±352.23)	-9324.63 (±1192.85)	-134.82 (±19.59)
{2,1,0,-1,-2}	WNS	-0.16 (±0.02)	-1.12 (±0.01)	-0.35 (±0.03)	-0.93 (±0.03)	-5.40 (±0.02)	-0.94 (±0.13)	-0.43 (±0.01)
	TNS	-102.78 (±19.86)	-2918.49 (±109.14)	-30.90 (±3.27)	-102.46 (±3.86)	-3416.78 (±672.95)	-10149.15 (±2620.79)	-132.31 (±17.93)
{1,0,-1,-2}	WNS	-0.13 (±0.01)	-1.07 (±0.01)	-0.32 (±0.01)	-0.85 (±0.01)	-5.37 (±0.01)	-0.59 (±0.03)	-0.39 (±0.03)
	TNS	-52.02 (±6.14)	-2602.25 (±34.66)	-27.69 (±0.65)	-84.83 (±1.73)	-2837.15 (±51.56)	-5250.15 (±216.33)	-134.75 (±29.44)

Table 7: Comparisons of TNS ($\times$ ns) and WNS ($\times$ ns) for macro placement derived by TP3D with different choices of K.

C.5 Comparison Study on Number of GNN Layers

We conducted an additional comparison study on the number of layers in the GNN architecture. As shown in Figure 11, the results demonstrate that the overall performance of the model remains robust under the GNN framework.

C.6 Visualization

We visualized the results of TP3D on all designs. We can see that TP3D clearly tends to place macro in edges and corners. This approach can fully reserve the central area for standard cells, which is more conducive to routing and thus achieves better timing results.

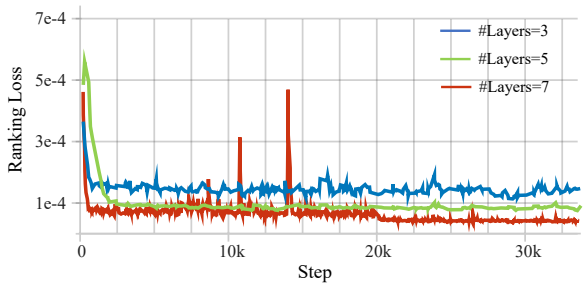
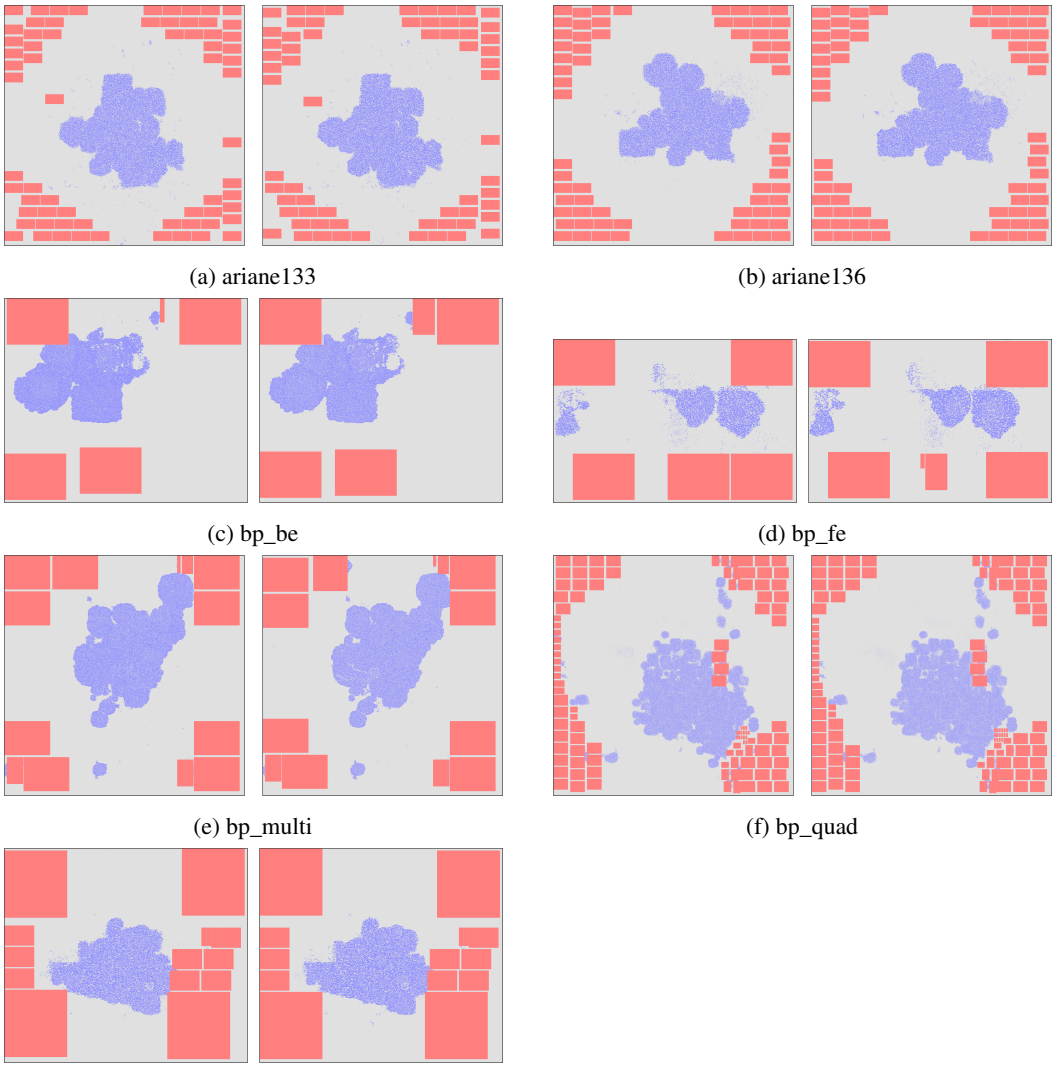


Figure 11: **Comparison study on number of GNN layers**. We present the training curves of the ranking loss with different numbers of GNN layers. The results are robust with our model architecture.



(a) ariane133 (b) ariane136

(c) bp_be (d) bp_fe

(e) bp_multi (f) bp_quad

(g) swerv_wrapper

Figure 12: The visualization results of global placement with TP3D

D LICENSES

We show the following open-source project used in this paper. We will also open-source our code once the paper is accepted.

- OpenROAD (Ajayi et al., 2019): BSD3-Clause License
- Open3DBench (Shi et al., 2025a): No License
- FlexPlanner (Zhong et al., 2024): No License
- Analytical-M3D (Zhao et al., 2025b): No Licenses