IncreMacro-3D: Incremental Macro Placement for Face-to-Face Stacked Memory-on-Logic 3D ICs

Lancheng Zou*, Sing Sen Ye*, Shuo Yin, Yuan Pu, Jiaxi Jiang, Siting Liu, Yuxuan Zhao, Bei Yu
The Chinese University of Hong Kong

***Abstract*—Face-to-face stacked 3D ICs, such as memory-on-logic (MoL) architectures, have emerged as a promising solution to overcome the limitations of traditional 2D integration by offering enhanced performance, power efficiency, and density. Given the increasing design complexity of modern system-on-chips (SoCs), achieving high-quality macro placement is critical, as it plays a decisive role in determining the final performance, power, and area (PPA) metrics. However, existing RTL-to-GDS 3D physical design flows for MoL 3D ICs rely heavily on manual macro placement, which becomes increasingly challenging and time-consuming for modern SoCs with a vast number of macros. In this paper, we introduce an innovative macro placement algorithm, IncreMacro-3D, which employs graph neural network-based macro repartitioning and 3D macro position refinement, thereby facilitating subsequent steps in 3D physical design flow. The experimental results on several benchmark circuits demonstrate that the proposed approach can reduce the routed wirelength, worst negative slack (WNS), total negative slack (TNS), and total power consumption by 6.1%, 44.2%, 62.8%, and 0.6% compared to state-of-the-art analytical placer for MoL 3D ICs.**

I. Introduction

As Moore's Law approaches its limits, 3D integrated circuits (3D ICs) have emerged as a transformative technology in modern chip design, with three primary integration approaches illustrated in Fig. 1: through-silicon vias (TSVs), monolithic, and face-to-face (F2F) stacked [1]. While TSV-based 3D stacking has been widely adopted, its relatively large geometric dimensions and low manufacturing yield limit its applicability to designs with a small number of inter-die connections, thereby constraining the full potential of advanced 3D integration [2]. Monolithic 3D integration, on the other hand, offers a promising alternative by fabricating 3D systems sequentially rather than stacking pre-fabricated 2D dies. This approach enables fine-grained vertical interconnections and mitigates the area overhead associated with TSVs. However, despite its advantages, monolithic 3D ICs currently face significant challenges in terms of manufacturing yield and cost [3]. F2F stacked 3D ICs comprise two prefabricated dies interconnected via hybrid bonding terminals (HBTs) situated on the topmost metal layer. The simplicity of the manufacturing process, coupled with the compact size of the bonding terminals, facilitates high integration density at a lower cost, establishing F2F bonding as a highly advantageous approach [4].

* These authors contributed equally to this paper.
This work is partially supported by The Research Grants Council of Hong Kong SAR (No. CUHK14211824 and No. CUHK14210723).

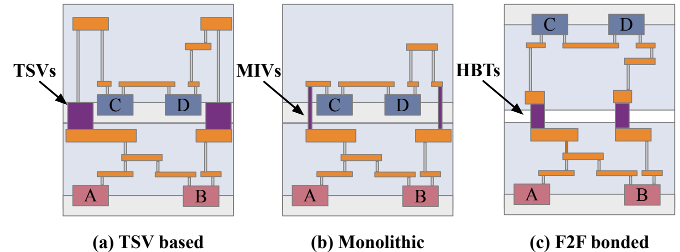


Fig. 1 Three types of integration techniques used in 3D ICs.

Memory-on-Logic (MoL) architecture is a paradigm in 3D ICs that vertically integrates logic and memory dies in a stacked configuration. By placing memory macros directly on top of the logic die, this approach minimizes the physical distance between memory and processing units, thereby significantly reducing data transfer latency and power consumption. Moreover, the increased memory bandwidth enabled by fine-grained vertical interconnections enhances performance for data-intensive applications, such as deep learning accelerators [5].

While 3D ICs offer significant advantages in terms of performance, power efficiency, and integration density, their realization requires a comprehensive consideration [6]. Shrunk-2D [7] represents the first approach to leverage commercial 2D electronic design automation (EDA) tools for the physical design of 3D ICs. In this methodology, standard cells and macros are scaled down to fit within a reduced 3D footprint. Subsequently, conventional 2D place-and-route (PnR) tools are employed to generate a pseudo-3D design. Following this, tier partitioning is performed to allocate logic cells to different dies. Finally, tier-by-tier routing is carried out using pre-determined 3D via locations. To address inaccuracies in RC parasitics estimation and the challenges associated with handling small geometries during PnR caused by shrinking, Compact-2D [8] performs PnR with interconnect RC scaling in a footprint that is twice the size of the 3D footprint to generate a pseudo-3D design. The cell locations are subsequently linearly mapped to their corresponding positions within the 3D footprint, followed by tier partitioning, post-tier-partitioning optimization, and incremental routing. Macro-3D [9] is a physical design methodology tailored specifically for MoL 3D ICs. Macros in top die are shrunk and their pin locations are preserved.

However, existing physical design flows [7]–[11] for 3D ICs heavily rely on manual macro placement and often lack further

978-3-9826741-1-7/DATE26/© 2026 EDAA

macro placement refinement. Given the significant impact of macro placement on the final power, performance, and area (PPA) outcomes, extensive efforts have been dedicated to refine macro placement in 2D ICs [12]–[16]. The lack of equivalent refinement techniques for 3D macro placement highlights a critical gap, as manual placement or suboptimal methods can result in increased interconnect delays, routability issue, and inefficient utilization of vertical integration. Consequently, addressing macro placement optimization in 3D ICs is essential to fully leverage the potential of 3D integration technologies. Especially in MoL 3D ICs, where macros can be placed on both dies but logic cells are restricted to the logic die, macro placement refinement presents unique challenges, including: (1) determining which macros should be assigned to the memory die, and (2) deciding whether different macro placement refinement strategies should be applied to different dies.

To address the aforementioned challenges, we propose a novel incremental macro placement framework for MoL 3D ICs, named IncreMacro-3D, to refine the initial 3D placement results. The primary objective of IncreMacro-3D is to achieve an improved macro partitioning outcome, avoiding entrapment in a local minimum caused by previous partitioning approaches that lack consideration of physical information. After partitioning optimization, a two-phase macro position refinement is conducted: macros in the logic die are pushed toward the chip boundary to improve routability, while macros in the memory die are moved closer to their connected logic cells to minimize data transfer delays.

Our major contributions are listed as follows:

- To the best of our knowledge, the proposed IncreMacro-3D is the first work to perform incremental macro placement for MoL 3D ICs based on the placement prototype by the analytical 3D placer.
- The proposed IncreMacro-3D framework consists of two main steps designed to fully exploit the unique characteristics and advantages of the MoL architecture: (i) graph neural network-based macro repartitioning and (ii) two-phase macro position refinement, which together enhance routability in the logic die and reduce wirelength between inter-die memory and logic.
- A comprehensive 3D physical design flow for MoL 3D ICs is implemented using commercial tools to evaluate the post-route PPA results.
- Experimental results across various benchmarks demonstrate that IncreMacro-3D outperforms the state-of-the-art analytical placer for MoL 3D ICs by reducing routed wirelength, worst negative slack (WNS), total negative slack (TNS), and total power consumption by 6.1%, 44.2%, 62.8%, and 0.6%, respectively.

II. Preliminaries

A. Macro Placement

Macro placement plays a critical role in achieving high quality of result (QoR). However, as the number of macros in modern SoCs continues to grow, it has become increasingly time-consuming and impractical for engineers to manually generate floorplan or macro placement. Consequently, extensive research has been conducted to develop automated algorithms aimed at improving macro placement quality. They can be classified into three typical types: (1) One-stage mixed-size placement: places macros and standard cells concurrently, e.g., NTUplace3 [17], SimPL [18], and MAPLE [19]. However, a significant challenge associated with this method is the legalization of large macros, which frequently leads to non-legalizable placement. (2) Three-stage mixed-size placement: this workflow consists of placement prototype, incremental macro placement, and standard cell placement, e.g., XDP [12], MP-tree [13], and CP-tree [15]. The prototyping stage begins by calculating initial positions for macros and standard cells without applying the non-overlapping constraint. Following this, a macro placer determines legal positions and orientations for the macros, optimizing key metrics such as wirelength, routability, and displacement. Lastly, with the macros fixed in their final positions, the standard cells are placed to ensure there are no overlaps between macros and standard cells. (3) Reinforcement learning-based macro placement: this approach uses reinforcement learning frameworks to train models that can adaptively learn placement strategies by interacting with the placement environment, e.g, MaskPlace [20], WireMask-BBO [21], and LAMPlace [22]. However, despite its promise, the reinforcement learning-based approach faces several challenges. One significant issue is the high computational cost associated with training RL models, as the placement environment often requires complex simulations that are time-consuming to evaluate. Additionally, achieving generalization across different design instances remains a challenge, as RL agents trained on specific layouts may struggle to perform well on unseen designs with varying characteristics.

For 3D ICs, the introduction of the vertical dimension presents new challenges for effective placement algorithms. ePlace-3D [23] presents an electrostatics based 3D density function to ensure global smoothness. Additionally, a bistratal wirelength model has been proposed to more accurately capture the wirelength of 3D nets [24]. The ICCAD 2023 contest [25] has further drawn significant attention from researchers, showcasing various 3D mixed-size placers [26], [27]. Different from these one-stage 3D mixed-size placer, K-POCS [28] is a macro placer for three-stage 3D mixed-size placer. The proposed method places macros sequentially into space tiles, optimizing a cost function that considers factors including wirelength, displacement, and distance between current location and geometric center of non-fully occupied tile. Open3D-DMP [29] is a placer tailored for MoL 3D ICs, which includes top-die macro placement, bottom-die macro placement, and standard cell placement. In top/bottom-die macro placement, the components from the opposite die are scaled to a minimal size while preserving pin locations, and DREAMPlace [30] is utilized to complete the placement process. During standard cell placement, the macros in the top-die are projected to bottom die again to perform pseudo-3D placement.

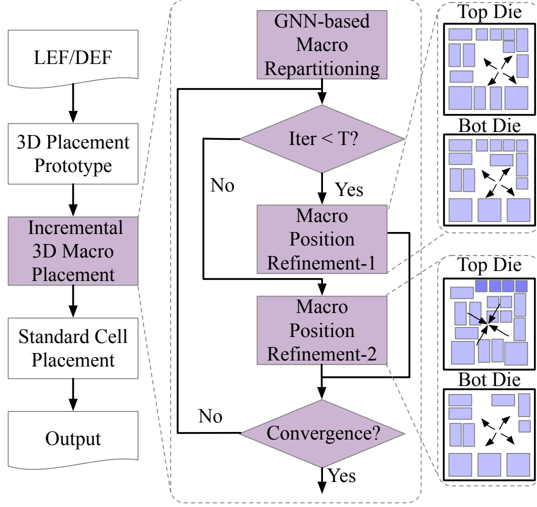


Fig. 2 The flow of 3D three-stage mixed-size placement and the proposed IncreMacro-3D workflow.

B. Problem Formulation

Similar to the three-stage mixed-size placement process in 2D ICs, the three-stage mixed-size placement in 3D ICs comprises three key stages: 3D placement prototyping, 3D macro placement, and 3D standard cell placement. For MoL 3D ICs, since standard cells are restricted to the bottom die, the final stage is simplified to 2D standard cell placement by projecting the top-die macros onto the bottom die. The proposed IncreMacro-3D method focuses on the second stage, specifically refining macro placement based on an initial 3D placement prototype.

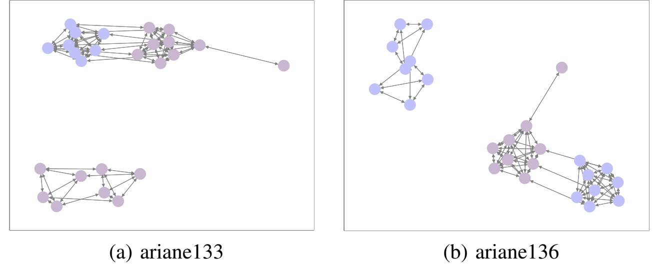
Given a netlist (V, E) where $V = \{v_1, \cdots, v_n\}$ is the node set, $E = \{v_1, \cdots, v_m\}$ is the net set, $C = \{c_1, \cdots, c_{n_c}\}$ is the standard cell set and $C \subset V$, $M = \{m_1, \cdots, m_{n_m}\}$ is the macro set and $M \subset V$, a 3D placement prototype contains initial cell positions $P_c = \{(x_1, y_1), \cdots, (x_{n_c}, y_{n_c})\}$, initial macro positions $P_m = \{(x_1, y_1), \cdots, (x_{n_m}, y_{n_m})\}$, and the initial tier assignment of macros $T = \{t_1, \cdots, t_{n_m}\}$ where t_i is a binary number (0/1 denotes the macro is placed on bottom/top die). The objective of the proposed IncreMacro-3D is to find a refined tier assignments and positions of macros, which can improve wirelength, displacement and routability to fully leverage the potential of MoL architecture.

III. Algorithm

Fig. 2 illustrates 3D three-stage mixed-size placement flow we used and the overall workflow of the proposed IncreMacro-3D. The framework comprises a graph neural network–based macro repartitioning stage followed by a two-phase macro position refinement. The 3D placement prototype is generated by an analytical MoL placer, i.e., Open3D-DMP [29].

A. GNN-based Macro Repartitioning

During the placement prototyping phase, area-balanced min-cut partitioning is utilized to allocate macros across different dies [29]. However, this approach fails to account for



(a) ariane133 (b) ariane136

Fig. 3 Clustering results derived from the learned graph representations for the two designs.

physical and hierarchical information, resulting in suboptimal partitioning results. Inspired by TP-GNN [31], a graph neural network (GNN)-based tier partitioning algorithm designed for standard cell partitioning, we adopt a GNN-based tier partitioning approach for macro repartitioning, incorporating both physical and hierarchical information.

Given the synthesized netlist, we construct a hierarchical graph, G_m, for macros. In this graph, each node represents a group of macros that share the same lowest-level hierarchy. For example, consider two macros named "top/module1/i1" and "top/module1/i2". Since both macros share the same lowest-level hierarchy, "top/module1", they are grouped into the same node in G_m. An edge is added between two nodes if any connections exist between their respective hierarchies. The initial node features include: number of macros, total number of pins, average number of nets, average macro area, and average number of connected standard cells. We use the message passing function to obtain the deep node representations, which is formulated as

$$\mathbf{y}_v^k = \sigma(\mathbf{W}^k \cdot (\mathbf{y}_v^{k-1} + \frac{1}{|N(v)|} \sum_{u \in N(v)} \mathbf{y}_u^{k-1})), \quad (1)$$

where σ is the sigmoid function, $N(v)$ is the neighbors of node v, $\mathbf{y}_v^k$ is the embedding of node v at level k, $\mathbf{y}_v^0$ denotes the initial node features of v, and $\mathbf{W}^k$ is the learnable weight parameters at level k. In the implementation, we employ a two-level message-passing mechanism with a hidden dimension of 32.

Moreover, an unsupervised learning approach is utilized for node representation learning, employing a cross-entropy-based loss function to encourage the embeddings of adjacent nodes to be more similar while ensuring that the embeddings of non-adjacent nodes remain distinct. The loss function is shown as follows:

$$\mathcal{L}_{\text{unsup}} = -\sum_{v \in G_m} \Bigg(\sum_{u \in N(v)} \log \left(w_{vu} \sigma(\mathbf{y}_v^\top \mathbf{y}_u) \right) + \sum_{n \in \text{Neg}(v)} \log \left(w_{vn} \sigma(-\mathbf{y}_v^\top \mathbf{y}_n) \right) \Bigg), \quad (2)$$

where $\mathbf{y}_v$ is the output node embedding of v after message passing, $\text{Neg}(v)$ denotes the non-adjacent nodes of v. w_{vu} is a weighted parameter to make the closer neighbor's embedding

be more similar, which is defined by $\frac{\max d_{v\cdot} - d_{vn}}{\max d_{v\cdot} - \min d_{v\cdot}} + 1$, where d_{vn} is the distance between the centroid of node v and u.

After training, we utilize K-means clustering to partition G_m based on the final node embeddings $\mathbf{y}$ of each node for tier assignment. The number of clusters is set to 2. Fig. 3 shows examples of clustering results. Once the two clusters are generated, the tier assignment is determined by the cumulative distance between the macros in each cluster and the centroid of the standard cells. The cluster with the smaller total distance is assigned to the top die, as it approximates stronger connectivity to standard cells, while the other cluster is assigned to the bottom die. This approach ensures that macro partitioning more effectively leverages the benefits of the MoL architecture.

B. Macro Position Refinement-1

After macro repartitioning, the first stage of macro position refinement focuses on prioritizing bottom-die routability while simultaneously optimizing wirelength. This is achieved by an iterative process including wirelength driven macro shifting and macro legalization.

1) Wirelength-driven Macro shifting: To facilitate the use of the gradient descent method, we adopt the weighted-average model [32] to estimate the wirelength cost, which is defined as

$$WA_{net} = \frac{\Sigma_{v_i \in net}\ x_i e^{\frac{x_i}{\gamma}}}{\Sigma_{v_i \in net}\ e^{\frac{x_i}{\gamma}}} - \frac{\Sigma_{v_i \in net}\ x_i e^{-\frac{x_i}{\gamma}}}{\Sigma_{v_i \in net}\ e^{-\frac{x_i}{\gamma}}}, \tag{3}$$

where γ regulates the balance between the accuracy and the smoothness of the approximation to the half-perimeter wirelength (HPWL).

Thus, the optimization objective of the first phase of macro position refinement can be defined by

$$\min \mathcal{L}_{phase1} = \sum_{e \in \text{Net}_{macro}} WA_e, \tag{4}$$

where Net_{macro} is the macro-related nets. After each optimization iteration, the position of macro m_i is updated by a displacement $(\Delta x_i, \Delta y_i)$.

2) Legalization: Following macro shifting, legalization process is applied to resolve macro overlaps while optimizing for routability.

$$\begin{aligned} \min\ & \sum_{i=1}^{|M|} \big(|x_i' - x_i| + |y_i' - y_i| \\ & + \min(x_i, W - x_i) + \min(y_i, H - y_i)\big) && (5) \\ \text{s.t. } & x_j' - x_i' \geq \frac{w_i + w_j}{2}, \forall e_{ij} \in G_h, && (6) \\ & y_j' - y_i' \geq \frac{h_i + h_j}{2}, \forall e_{ij} \in G_v, && (7) \\ & \frac{w_i}{2} \leq x_i' \leq W - \frac{w_i}{2}, \frac{h_i}{2} \leq y_i' \leq H - \frac{h_i}{2}, \forall m_i \in M, && (8) \end{aligned}$$

where (x_i, y_i) and (x_i', y_i') represent the original position and the legalized position of macro m_i. G_h and G_v denote the horizontal and vertical constraint graphs, as introduced in XDP [12]. The objective is to position macros as close to the chip boundary as possible while minimizing their displacement from their original locations.

3) Incremental Partitioning for Routability Optimization: Upon the completion of the first stage, the regularity of the top-die and bottom-die macros is established, and the macros are successfully pushed toward the chip boundary.

To further improve the routability in bottom die, where all standard cells are placed, we employ an incremental partitioning for routability optimization. We first calculate the density of bottom die, if it exceeds the pre-determined density threshold T_d, a portion of the bottom-die macros is moved to the top die to leave more central space and available routing resource for standard cell placement and routing. Macros positioned near the chip boundary that result in the minimal addition of 3D nets are selected. We define top-die macros originally assigned as **logic-affinity macros**, while those assigned through incremental partitioning are defined as **logic-disaffinity macros**.

C. Macro Position Refinement-2

At this stage, the optimization prioritizes the placement of top-die macros. With the bottom-die macros already positioned along the chip boundary in an organized manner, the central space available for standard cell placement can be estimated with greater precision. This is especially critical for logic-affinity macros due to their stronger connections with standard cells. To reduce the length of inter-die memory and logic connections, these macros should be placed directly above the region of standard cells.

1) Macro Shifting for MoL: We propose a convergence-based shift algorithm to reposition the logic-affinity macros to a convergence point, (p_x, p_y), where its surrounding is macro-free under the top-die. This accommodates direct vertical connections between bottom-die cells and the logic-affinity macros. Conversely, the algorithm is not applied to logic-disaffinity macros to prioritize the placement of logic-affinity macros. Logic-disaffinity macros are maintained in close proximity to their associated bottom-die macros, which need to remain positioned near the chip boundary. To achieve this, we define a convergence loss for logic-affinity macro m_i as

$$\mathcal{L}_{converg}(m_i) = (p_x - x_{m_i})^2 + (p_y - y_{m_i})^2, \tag{9}$$

where (p_x, p_y) is the position of convergence point. Using this loss, the logic-affinity macros will be guided towards the convergence point. Thus, it is crucial to identify a point that can accurately approximate the centroid of the final standard cell placement.

Using the chip center as the convergence point may result in biased guidance, as the bottom-die macros are unlikely to be uniformly distributed along the chip boundary. To address it, we obtain the convergence point by calculating the density of bottom-die macro in different regions to obtain horizontal and

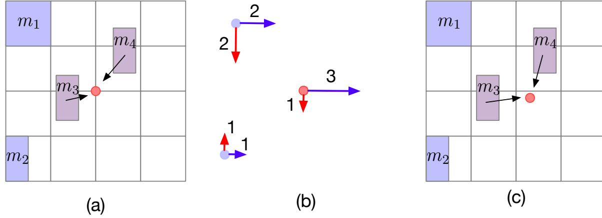


Fig. 4 Visualization of convergent-point adjustment. Blue and purple blocks represent the bottom-die and top-die macros. In (b), the red point represent resultant offset for the top-die center. The blue arrow denotes the offset in x-direction while the red arrow denotes the offset in y-direction induced by blue blocks.

vertical offsets on center, c_x and c_y, which can be expressed as:

$$c_x = \sum_{i}^{|M_{bot}|} d_{x_i} \times r_i \times W, r_i = \frac{w_{m_i} h_{m_i}}{WH}, \tag{10}$$

where W and H are the width and height of the chip, m_i is the ith bottom-die macro, w_{m_i} and h_{m_i} denote its width and height, and d is a sign number (± 1) to represent the direction of adjustment which toward to the center. Thus $p_x = \frac{1}{2}W + c_x$, c_y and p_y are obtained in a similar way. Fig. 4 presents the design chip layout with dimension of 32×32.

Overall, the objective of macro shifting in phase 2 is shown as

$$\min \mathcal{L}_{phase2} = \sum_{e \in \text{Net}_{macro}} WL_e + \alpha \sum_{m_i \in M_{lam}} \mathcal{L}_{converg}(m_i), \tag{11}$$

where M_{lam} is the set of top-die logic-affinity macros.

2) Legalization: The legalization of logic-affinity macros use only displacement as the objective, but the remaining macros including logic-disaffinity macros still use the same legalization process as shown in Equations (5) to (8).

IV. Flow Implementation

In this section, we outline our methodology for extending traditional 2D commercial tools to facilitate physical design for MoL 3D ICs. The proposed flow eliminates the need for time-consuming manual macro placement and addresses the limitations of suboptimal results produced by one-stage mixed-size placement.

A. Handling Technology Files

To represent an appropriate 3D design, we first create a 3D process design kit (PDK) based on Nangate45 [33]. To maintain the validity of the 2D PnR results for the 3D design while accurately capturing net parasitics, the back-end-of-line (BEOL) of the 3D metal stack is modeled using a captable file for parasitic extraction and a techlef file for layer abstraction. In the metal layers of the top die, a suffix "_top" is appended to their names for differentiation, such as M1_top. The properties of M1_top are identical to those of M1 in the bottom die, allowing the metal layers in the top and bottom dies to be

TABLE I The statistics of MacroPlacement benchmarks [34].

Benchmark	# Macros	# Cells	# Nets	Frequency / MHz	Area / mm^2
ariane133	133	99996	123924	769.23	1.00
ariane136	136	106395	130202	769.23	1.00
bp_quad	220	646578	876749	769.23	4.52
mempool_tile	20	95554	109369	165.00	0.36
NVDLA	128	152770	207807	1111.11	2.25

considered mirrored. The primary difference lies in the routing direction, which is adjusted to preserve the interleaving of metal layers. In our implementation, both the bottom die and the top die are designed with six metal layers each. Moreover, the F2F bonding is included as a special via between M6 and M6_top in our case. The pitch, width, and resistance of the F2F bonding is set as 1μm, 1μm, and 4Ω.

To integrate the tier information of top-die macros into commercial 2D tools, the LEF files for the top-die macros are generated by duplicating the original LEF files and appending the suffix "_top" to the macro names. For example, if the original macro is named macro1, the correspoending top-die macro is named macro1_top. To prevent placement blockages caused by top-die macros, their sizes are scaled down to the minimum size, equivalent to the size of the smallest filler cell. Furthermore, the pin locations are preserved to serve as anchors for standard cell placement. Notably, the pin layers of top-die macros are mapped to the top-die metal layers in a mirrored manner to ensure accurate 3D routing. For instance, if pin A of macro1 is located on layer M2, then pin A of macro1_top will correspondingly be located on layer M2_top.

B. Placement and Routing for MoL 3D ICs

Given the macro placement result from Section III, we use the 2D placement engine in commercial tool to finish standard cell placement of the third stage in three-stage mixed-size placement for MoL 3D ICs. Prior to placement, in-house scripts are employed to parse the placement results, including tier assignment information, and subsequently generate the corresponding 3D netlist and DEF file. Then, the 3D netlist and DEF file are fed into commercial tools to perform placement and routing.

V. Experiments

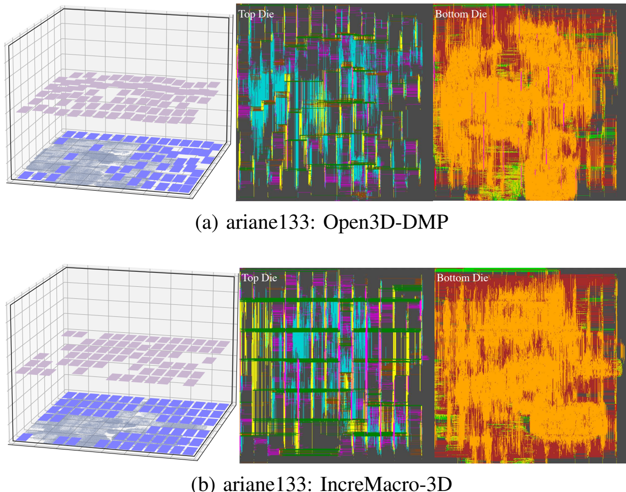
A. Experimental Setup

We implement IncreMacro-3D in Python, utilizing DGL [35] for graph learning, and PyTorch [36] for gradient computation and backpropagation. Additionally, Gurobi [37] is employed as the ILP solver. We run IncreMacro-3D on a Linux machine with 32-core Intel Xeon Gold 6226R CPU (2.90GHz), one NVIDIA GeForce RTX 3090 GPU, and 128GB RAM.

Five designs from MacroPlacement [34] are selected as benchmarks for our evaluation. These designs are synthesized using Cadence Genus with the Nangate45 [33] PDK. The post-synthesis statistics of the evaluated benchmarks are shown in TABLE I. We implement the 3D physical design flow described in Section IV with Cadence Innovus as our placement and routing engines.

TABLE II Comparison of post-route results among Open3D-DMP [29], IncreMacro-3D w.o. GMRP, and IncreMacro-3D.

Benchmark	Open3D-DMP [29]					IncreMacro-3D w.o. GMRP					IncreMacro-3D				
	rWL (um)	WNS (ns)	TNS (ns)	Power (mW)	Runtime (s)	rWL (um)	WNS (ns)	TNS (ns)	Power (mW)	Runtime (s)	rWL (um)	WNS (ns)	TNS (ns)	Power (mW)	Runtime (s)
ariane133	4394142	-0.070	-33.120	840.96	80	**4074873**	-0.063	-4.553	**836.54**	416	4134944	**-0.038**	**-4.186**	846.03	437
ariane136	4694011	-0.056	-20.710	866.86	85	**4383761**	-0.044	-12.693	**853.16**	432	5051825	**-0.036**	**-5.510**	858.79	455
bp_quad	30365842	-0.687	-11649.900	4511.72	325	27993175	-0.516	-8499.1	4490.81	742	**25876519**	**-0.465**	**-6595.600**	**4455.67**	795
mempool_tile	2538213	-0.021	-1.361	79.66	110	2578753	-0.093	-27.690	80.82	340	**2467694**	**-0.019**	**-1.225**	**79.55**	378
NVDLA	11106336	-0.143	-6.136	2368.23	70	9810449	-0.009	-0.010	2345.79	149	**9500393**	**-0.003**	**-0.011**	**2337.83**	159
Average	1.000	1.000	1.000	1.000	**1.000**	**0.936**	1.386	4.365	0.996	3.103	0.939	**0.558**	**0.372**	**0.994**	3.319



(a) ariane133: Open3D-DMP

(b) ariane133: IncreMacro-3D

Fig. 5 Visualization of placement and routing results for ariane133 by Open3D-DMP and IncreMacro-3D. Purple rectangles denote top-die macros, blue rectangles represent bottom-die macros, and the standard cells on bottom die are in gray.

We generate the 3D placement prototypes of all benchmark circuits using Open3D-DMP [29], which is an analytical MoL 3D placer. Given a 3D placement prototype, we employ the proposed IncreMacro-3D to refine the 3D macro placement. Finally, post-route results are evaluated using the implemented 3D physical design flow. Moreover, to evaluate the effectiveness of GNN-based macro repartitioning, we conduct an ablation study by performing IncreMacro-3D without GNN-based macro repartitioning. This variant is referred to as "IncreMacro-3D w.o. GMRP" in the subsequent sections.

B. Comparison with Existing MoL 3D Placer

We report the post-route results including routed wirelength (rWL), worst negative slack (WNS), total negative slack (TNS), and power consumption to assess the effectiveness of the proposed framework.

In TABLE II, the detailed results, including post-route outcomes and runtimes, are presented for a comprehensive comparison. The runtimes for both IncreMacro-3D w.o. GMRP and IncreMacro include the time required for placement prototyping (i.e., Open3D-DMP).

In terms of timing performance, IncreMacro-3D consistently demonstrates superior results, achieving reductions in WNS and TNS by 44.2% and 62.8%, respectively, compared to Open3D-DMP on average. Moreover, IncreMacro-3D reduces the rWL and power consumption by 6.1% and 0.6%

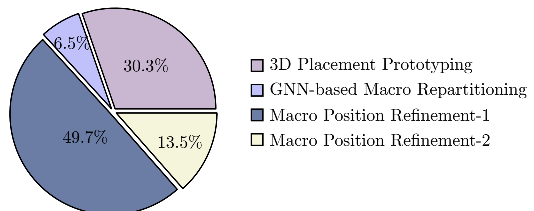


Fig. 6 Average runtime breakdown of 3D placement prototyping by Open3D-DMP and subsequent IncreMacro-3D for 3D macro placement refinement.

compared to the baseline. The results of IncreMacro-3D w.o. GMRP highlight the critical role of physical information-aware repartitioning in MoL macro refinement, as it helps prevent macro position optimization from becoming trapped in suboptimal solutions.

We also provide visualization of the placement results and post-route layouts for ariane133 in Fig. 5. It's observed that the IncreMacro-3D is able to leave enough space solely for standard cell placement for improved routability. Moreover, top-die macros are successfully guided to move towards the position of center region of standard cells to reduce the lengths of inter-die memory and logic cells connections.

The runtime breakdown is shown in Fig. 6. 3D placement prototyping takes about one third of the Open3D-DMP+IncreMacro-3D workflow. In IncreMacro-3D, the runtime overhead of GNN-based macro repartitioning is minimal but it is crucial for the 3D macro refinement. Although IncreMacro-3D incurs nearly 2× the runtime overhead, this increase is justified by the improved timing performance it achieves. Moreover, the overhead is negligible compared to the significantly longer runtime required for standard cell placement using commercial tools (e.g., ranging from 30 minutes to 6 hours in the evaluated benchmarks).

VI. Conclusion

In this paper, we present IncreMacro-3D, a novel framework for macro placement refinement for MoL 3D ICs. To fully exploit the advantages of MoL 3D ICs, the proposed workflow first repartitions macros using graph learning based on initial physical information, followed by a two-phase 3D macro position refinement. This approach enhances routability and reduces the inter-die connection length between memory and logic components. Experimental results demonstrate that the proposed framework outperforms the existing MoL placer in terms of routed wirelength, timing, and power consumption.

REFERENCES

[1] E. Beyne, "The 3-d interconnect technology landscape," *IEEE Design & Test*, vol. 33, no. 3, pp. 8–20, 2016.

[2] M. Motoyoshi, "Through-silicon via (tsv)," *Proceedings of the IEEE*, vol. 97, no. 1, pp. 43–48, 2009.

[3] P. Batude, B. Sklenard, C. Fenouillet-Beranger, B. Previtali, C. Tabone, O. Rozeau, O. Billoint, O. Turkyilmaz, H. Sarhan, S. Thuries *et al.*, "3d sequential integration opportunities and technology optimization," in *IEEE International Interconnect Technology Conference*, 2014, pp. 373–376.

[4] T. Song, A. Nieuwoudt, Y. S. Yu, and S. K. Lim, "Coupling capacitance in face-to-face (f2f) bonded 3d ics: Trends and implications," in *IEEE Electronic Components and Technology Conference*, 2015, pp. 529–536.

[5] C. Li, Y. Yin, X. Wu, J. Zhu, Z. Gao, D. Niu, Q. Wu, X. Si, Y. Xie, C. Zhang *et al.*, "H2-llm: Hardware-dataflow co-exploration for heterogeneous hybrid-bonding-based low-batch llm inference," in *Proceedings of the 52nd Annual International Symposium on Computer Architecture*, 2025, pp. 194–210.

[6] Y. Zhao, L. Zou, and B. Yu, "Physical design for advanced 3d ics: Challenges and solutions," in *ACM International Symposium on Physical Design (ISPD)*, 2025, pp. 209–216.

[7] S. Panth, K. Samadi, Y. Du, and S. K. 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 (TCAD)*, vol. 36, no. 10, pp. 1716–1724, 2017.

[8] B. W. Ku, K. Chang, and S. K. Lim, "Compact-2d: A physical design methodology to build two-tier gate-level 3-d ics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 6, pp. 1151–1164, 2019.

[9] L. Bamberg, A. García-Ortiz, L. Zhu, S. Pentapati, D. E. Shim, and S. K. Lim, "Macro-3d: A physical design methodology for face-to-face-stacked heterogeneous 3d ics," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2020, pp. 37–42.

[10] P. Vanna-Iampikul, C. Shao, Y.-C. Lu, S. Pentapati, Y. Heo, J.-S. Choi, and S. K. Lim, "Snap-3d: A constrained placement-driven physical design methodology for high performance 3-d ics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 42, no. 7, pp. 2331–2335, 2022.

[11] S. S. K. Pentapati, K. Chang, V. Gerousis, R. Sengupta, and S. K. Lim, "Pin-3d: A physical synthesis and post-layout optimization flow for heterogeneous monolithic 3d ics," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020, pp. 1–9.

[12] J. Cong and M. Xie, "A robust detailed placement for mixed-size ic designs," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2006, pp. 188–194.

[13] T.-C. Chen, P.-H. Yuh, Y.-W. Chang, F.-J. Huang, and D. Liu, "Mp-trees: A packing-based macro placement algorithm for mixed-size designs," in *ACM/IEEE Design Automation Conference (DAC)*, 2007, pp. 447–452.

[14] H.-C. Chen, Y.-L. Chuang, Y.-W. Chang, and Y.-C. Chang, "Constraint graph-based macro placement for modern mixed-size circuit designs," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2008, pp. 218–223.

[15] Y.-F. Chen, C.-C. Huang, C.-H. Chiou, Y.-W. Chang, and C.-J. Wang, "Routability-driven blockage-aware macro placement," in *ACM/IEEE Design Automation Conference (DAC)*, 2014, pp. 1–6.

[16] Y. Pu, T. Chen, Z. He, C. Bai, H. Zheng, Y. Lin, and B. Yu, "Incremacro: Incremental macro placement refinement," in *ACM International Symposium on Physical Design (ISPD)*, 2024, pp. 169–176.

[17] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, "Ntuplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 27, no. 7, pp. 1228–1240, 2008.

[18] M.-C. Kim, D.-J. Lee, and I. L. Markov, "Simpl: An effective placement algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 31, no. 1, pp. 50–60, 2011.

[19] M.-C. Kim, N. Viswanathan, C. J. Alpert, I. L. Markov, and S. Ramji, "Maple: Multilevel adaptive placement for mixed-size designs," in *ACM International Symposium on Physical Design (ISPD)*, 2012, pp. 193–200.

[20] Y. Lai, Y. Mu, and P. Luo, "Maskplace: Fast chip placement via reinforced visual representation learning," *Conference on Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 24 019–24 030, 2022.

[21] Y. Shi, K. Xue, S. Lei, and C. Qian, "Macro placement by wire-mask-guided black-box optimization," *Conference on Neural Information Processing Systems (NeurIPS)*, vol. 36, pp. 6825–6843, 2023.

[22] H. Gu, J. Gu, K. Peng, Z. Zhu, N. Xu, X. Geng, and J. Yang, "Lamplace: Legalization-aided reinforcement learning based macro placement for mixed-size designs with preplaced blocks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2024.

[23] J. Lu, H. Zhuang, I. Kang, P. Chen, and C.-K. Cheng, "eplace-3d: Electrostatics based placement for 3d-ics," in *ACM International Symposium on Physical Design (ISPD)*, 2016, pp. 11–18.

[24] P. Liao, Y. Zhao, D. Guo, Y. Lin, and B. 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 (TCAD)*, vol. 43, no. 6, pp. 1624–1637, 2023.

[25] K.-S. Hu, H.-Y. Chi, I.-J. Lin, Y.-H. Wu, W.-H. Chen, and Y.-T. Hsieh, "2023 iccad cad contest problem b: 3d placement with macros," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2023, pp. 1–6.

[26] Y.-J. Chen, C.-H. Hsieh, P.-H. Su, S.-H. Chen, and Y.-W. Chang, "Mixed-size 3d analytical placement with heterogeneous technology nodes," in *ACM/IEEE Design Automation Conference (DAC)*, 2024, pp. 1–6.

[27] Y. Zhao, P. Liao, S. Liu, J. Jiang, Y. Lin, and B. Yu, "Analytical heterogeneous die-to-die 3d placement with macros," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2024.

[28] J.-M. Lin, P.-C. Lu, H.-Y. Lin, and J.-T. Tsai, "A novel blockage-avoiding macro placement approach for 3d ics based on pocs," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2022, pp. 1–7.

[29] Y. Shi, C. Gao, W. Ren, S. Xu, K. Xue, M. Yuan, C. Qian, and Z.-H. Zhou, "Open3dbench: Open-source benchmark for 3d-ic backend implementation and ppa evaluation," *arXiv preprint*, 2025.

[30] Y. Lin, S. Dhar, W. Li, H. Ren, B. Khailany, and D. Z. Pan, "Dreamplace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement," in *ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[31] Y.-C. Lu, S. S. K. Pentapati, L. Zhu, K. Samadi, and S. K. Lim, "Tp-gnn: A graph neural network framework for tier partitioning in monolithic 3d ics," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[32] M.-K. Hsu, V. Balabanov, and Y.-W. 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 (TCAD)*, vol. 32, no. 4, pp. 497–509, 2013.

[33] "Nangate technology node," https://si2.org/open-cell-library/.

[34] C.-K. Cheng, A. B. Kahng, S. Kundu, Y. Wang, and Z. Wang, "Assessment of reinforcement learning for macro placement," in *ACM International Symposium on Physical Design (ISPD)*, 2023, pp. 158–166.

[35] "DGL," https://www.dgl.ai/.

[36] "PyTorch," https://pytorch.org/.

[37] "Gurobi," https://www.gurobi.com/.