Partitioning-free 3D-IC Floorplanning

Shuo Ren, Zhen Zhuang, Rongliang Fu, Leilei Jin, Libo Shen, Bei Yu, Tsung-Yi Ho
The Chinese University of Hong Kong

***Abstract*—Although 3D IC integration offers a promising path to alleviate interconnect bottlenecks in 2D designs, efficient 3D floorplanning remains challenging due to its increased spatial complexity. Prior approaches that directly extend 2D representations into 3D suffer from exponential solution spaces, while pre-partitioning strategies constrain the global optimization landscape by fixing block-to-die assignments early. As a result, both approaches not only hinder comprehensive exploration of the 3D design space, but also overlook critical 3D, specific characteristics such as inter-die communication latency, which directly impact system performance but are often abstracted away in simplified 2D-extended models. To address these challenges, we propose GREAT3D, a partitioning-free 3D floorplanning framework that jointly optimizes block floorplan and die assignment without relying on predefined die partitioning. By a two-stage 3D SDP optimization and 2D refinement, GREAT3D effectively minimizes wirelength and latency under outline constraints. Experimental results on GSRC benchmarks show that Great3D consistently achieves lower wirelength than the baselines, with up to 60% reduction on large-scale designs. Furthermore, the method maintains competitive runtime performance while demonstrating better scalability and robustness across diverse benchmark sizes. These results establish Great3D as a scalable and effective partitioning-free solution for high-quality 3D IC floorplanning.**

***Index Terms*—3D IC, 3D Floorplanning, Partitioning-free, Semi-definite Programming**

I. INTRODUCTION

While transistor scaling has upheld Moore's Law, interconnect scaling lags behind, making routing delay the key bottleneck in modern two-dimensional (2D) integrated circuits (ICs) [1]. To address this, three-dimensional (3D) integration stacks device layers vertically, transforming long intra-die wires into short inter-die connections and increasing integration density [2]. Advances in fine-pitch hybrid bonding enable TSV-free, low-parasitic face-to-face (F2F) interconnects, accelerating 3D IC adoption in commercial designs [3]–[5]. These benefits, however, come with new physical-design challenges: designers must manage inter-die connectivity with non-uniform communication costs and explore an explosively larger placement solution space [2]. Among all backend stages, floorplanning sets the relative positions and die assignments of blocks, which dominate wirelength and communication latency in 3D stacks. Consequently, efficient 3D floorplanning has become a critical problem in modern IC design.

Unlike 2D floorplanning, which determines block locations within a single planar layer, 3D floorplanning must simultaneously determine both block location and die assignment across multiple vertically stacked tiers. This vertical dimension introduces fundamental new complexities, including z-axis feasibility, cross-tier density legality, and non-uniform communication latency. In particular, inter-die links typically

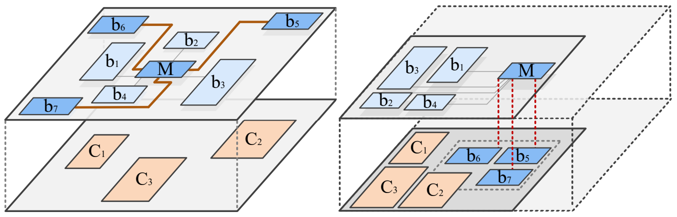


Fig. 1 Comparison of 3D floorplans before and after inter-die optimization. Our method (right) better utilizes vertical links and achieves reduced area under identical outline constraints.

incur higher delays than intra-die wires due to bonding and stack-level constraints, which must be accounted for during physical layout. Hence, existing 2D floorplanning methods are not directly applied to 3D floorplanning.

To address these 3D-specific challenges, several 3D floorplanning methods have been proposed in recent literature. Existing 3D floorplanning methods can be grouped into two categories, each making restrictive assumptions that limit their scalability and solution quality. The first category [6]–[9] extends 2D representations into 3D by embedding die assignment directly into the placement structure. For example, 3D slicing trees [6] enforce hierarchical layer partitioning, while grouped sequence pairs [7] encode cross-layer placement. FlexPlanner [9] uses a graph-based representation trained on prior layout data, but its performance heavily depends on the quality and generalizability of the training dataset, which limits its applicability to unseen architectures. Although these methods retain flexibility, they suffer from exponential solution space growth and entangled optimization objectives. As a result, they rely on slow metaheuristics and cannot easily decouple the die assignment from physical placement to reason about inter-die latency.

The second category [10]–[12] adopts a partitioning-first paradigm. These methods decompose the 3D task into sequential 2D subproblems through pre-processing. Using such as Fiduccia Mattheyses (FM) [12], [13] and spectral partitioning [14] for efficient block-to-die partitioning, they apply 2D floorplanning on each die and stack the results. This improves scalability but imposes structural constraints, disconnecting die assignment from floorplan optimization and preventing joint reasoning about inter-die connectivity and latency minimization. Therefore, these methods still fall short of fully addressing the complexities introduced by vertical stacking, which is the most important structure in 3D ICs.

To address these problems, we propose GREAT3D, a unified framework that jointly optimizes block floorplan and die assignment without partitioning. Unlike prior work, GREAT3D directly explores native 3D solutions without pre-processing or restrictive intermediate formats. As illustrated in Fig. 1, baseline layouts (left) constrain highly connected modules to a single tier, which limits vertical integration and leads to inefficient floorplan utilization. Our adjustment (right) remaps these nets across tiers, enabling shorter inter-die connections and better module distribution. These limitations highlight the need for partitioning-free, holistic 3D optimization. Overall, our contributions are as follows:

- We introduce a native-3D floorplanning method that jointly performs block floorplanning and die assignment within a unified optimization framework.
- Our multi-objective formulation balances interconnect cost and die utilization, producing compact layouts with reduced wirelength compared to state-of-the-art.
- The framework integrates latency-aware modeling, supporting early-stage performance estimation and design space exploration at the system level.

II. Preliminaries

A. 3D IC

Modern 3D integration falls into three categories [15]–[17]: (1) TSV-based, (2) monolithic, and (3) face-to-face (F2F) hybrid bonding. TSV-based integration [18] suffers from parasitics and area overhead, suiting sparse inter-die connections. Monolithic 3D ICs [19] require sequential processing, increasing cost and reducing yield. F2F bonding [20]–[22] offers high-density, low-parasitic interconnects by aligning pre-fabricated dies at the top metal layers. It supports die heterogeneity and reuse, gaining traction in modern 3D ICs [23], [24]. Our proposed GREAT3D naturally applies F2F-bonded systems and to quantify performance, we adopt a CPI-based latency model [25]. The minimize the average latency, the objective is defined as Equation (1).

$$\mathcal{H}(\mathbf{S}, L, \mathrm{MP}) = \mathrm{CPI}_{\mathrm{sta}} + \alpha_1 \cdot (\mathrm{lat}_{\mathrm{ppi}} - \mathrm{lat}_{\mathrm{sta}}), \tag{1}$$

where the latency $\mathrm{lat}_{\mathrm{ppi}} = \langle \mathbf{S}, L \rangle$ is the Frobenius inner product between the structure matrix $\mathbf{S} = \sum_p N_p$ and latency matrix $L \in \mathbb{R}^{n \times n}$. Here, N_p is the connectivity matrix for net e_p. Static CPI is modeled as $\mathrm{CPI}_{\mathrm{sta}} = \alpha_2 \cdot \mathrm{MP} + \beta_1$; delay as $\beta_2 \cdot \mathrm{wirelength}_{ppi}$, where MP models system-level constraints.

B. 3D Floorplan

In floorplanning, communication cost is commonly estimated using the half-perimeter wirelength (HPWL) model. In 2D, the HPWL of a net e_i connecting a set of blocks S_i is defined as Equation (2), which is widely adopted due to its simplicity and effectiveness [26], [27]. In practical optimization frameworks, multi-pin nets are commonly decomposed into pairwise interactions to yield differentiable surrogate objectives [26]. We adopt such a pairwise formulation, defining a connectivity matrix A^o, where each entry A^o_{ij} represents the aggregated connection strength between blocks i and j. This yields the objective function expressed as the inner product between connectivity and distance matrices as Equation (3).

$$\mathrm{WL}_{2D}(e_i) = \max_{b \in S_i}(x_b) - \min_{b \in S_i}(x_b) + \max_{b \in S_i}(y_b) - \min_{b \in S_i}(y_b), \tag{2}$$

$$\mathcal{W}(A^o, D) = \langle A^o, D \rangle = \sum_{i,j} A^o_{ij} \cdot D_{ij}. \tag{3}$$

where D_{ij} denotes the geometric square distance between blocks i and j. In 3D floorplanning, we naturally extend this geometric interpretation into 3D Euclidean space.

C. Motivation

Unlike traditional 2D floorplanning tasks, 3D IC design introduces heterogeneous communication patterns due to vertical integration. Intra-die communication is short and dense, whereas inter-die communication typically incurs higher cost and is constrained by bonding technology. Prior works [7], [11] often treat these two communication types uniformly by extending the 2D HPWL metric into 3D space.

However, such formulations rely on discrete z-coordinate values that serve more as die-level indicators than physical geometric distances. To better reflect this heterogeneity, we adopt a dual-level modeling strategy: we use 2D HPWL over (x, y) coordinates to capture detailed intra-die placement quality, and incorporate inter-die communication cost explicitly through a CPI-based system-level latency model [25]. These two components are then unified into a single objective via a weighted connection-distance formulation in Section II-D, enabling joint optimization of physical floorplan compactness and architectural communication latency.

D. Problem Formulation

Input:

- Netlist (V_b, V_p, E): $V_b = \{b_1, \ldots, b_n\}$ contains blocks; $V_p = \{p_1, \ldots, p_m\}$ are fixed pins; $E = \{e_1, \ldots, e_k\}$ contains nets each connecting blocks and/or pins.
- Block parameters: Width w_i, height h_i, area a_i, center (x_i, y_i, z_i); blocks must satisfy $r_i^{\min} \le w_i/h_i \le r_i^{\max}$.
- Design constraints: Bounding box (W, H), overlap tolerance $\epsilon_{\mathrm{overlap}}$, optional F2F bonding regions $\mathcal{B}_i$.

Output:

- 3D floorplan (x_i, y_i, z_i) of each block b_i: x_i, y_i are continuous planar coordinates; z_i is the discrete die assignment.
- Floorplan dimensions (W, H) per die and legal block sizes (w_i, h_i) satisfying aspect ratio constraints.
- Net partitioning: $E_{\mathrm{top}}, E_{\mathrm{bot}}, E_{\mathrm{inter\text{-}die}}$ induced by die assignments.
- Inter-die connection statistics: total count $|E_{\mathrm{inter\text{-}die}}|$ and computed density ρ_{cur}.

Constraints:

- Overlap: To ensure physical feasibility, blocks must not overlap beyond a predefined tolerance $\epsilon_{\mathrm{overlap}}$, where $\mathrm{Overlap}(b_i, b_j)$ measures the overlapping area between blocks b_i and b_j.

$$\sum_{i<j} \mathrm{Overlap}(b_i, b_j) \le \epsilon_{\mathrm{overlap}}, \tag{4}$$

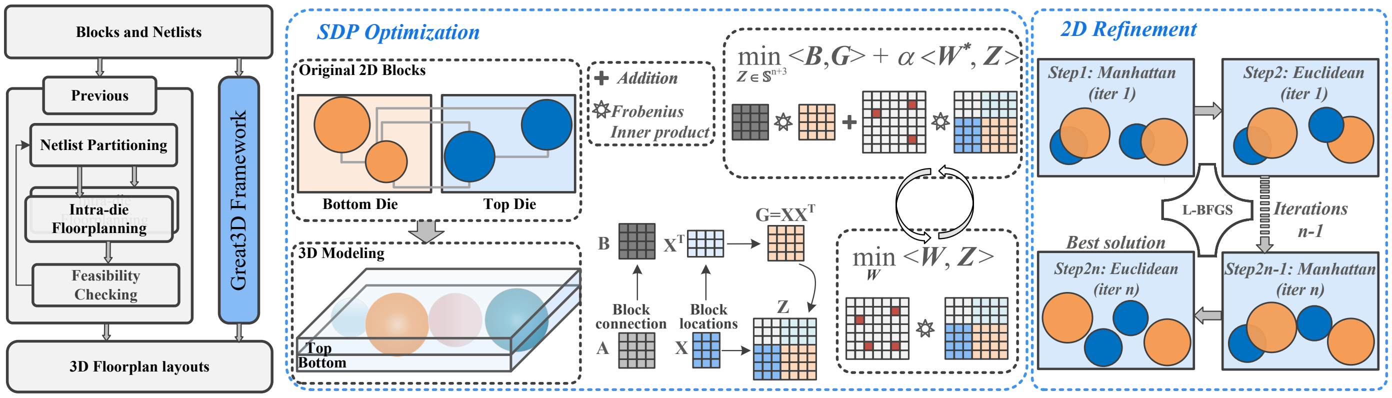


Fig. 2 Overview of the proposed GREAT3D optimization framework shown in Section III. The framework consists of two key components: SDP-based optimization (Section III-A) and 2D refinement (Section III-B).

- Aspect ratio: Soft blocks must maintain specified aspect ratio bounds denoted by $r_i^{\min}$ and $r_i^{\max}$:
$$r_i^{\min} \le \frac{w_i}{h_i} \le r_i^{\max}, \quad \forall b_i \in B_{\text{soft}}. \tag{5}$$
- Fixed-outline: All blocks must reside within the layout region, denoted by W, H, and both top die and bottom die should have the same outline:
$$\begin{aligned} 0 \le x_i - \frac{w_i}{2}, \quad & x_i + \frac{w_i}{2} \le W, \\ 0 \le y_i - \frac{h_i}{2}, \quad & y_i + \frac{h_i}{2} \le H, \quad \forall b_i \in B. \end{aligned} \tag{6}$$
- Hybrid bonding density: To capture this unique 3D characteristic, we explicitly impose a bonding density constraint that limits the number of vertical connections per unit area.
$$|E_{\text{inter-die}}| \le \rho_{\max} \cdot area_{\text{die}}. \tag{7}$$
where $\rho_{\max}$ denotes the maximum allowable inter-die connection density under specific manufacture standards.

Objective:

Faced with the inputs and constraints mentioned above, we need to take both the floorplan metrics and the characteristics of 3D IC into consideration. Then a direct trade-off formulation method can be used as follows:
$$\min_{\Theta} \quad \mathcal{F}^o = (1-\mu)\cdot \mathcal{W}(A^o, D) + \mu \cdot \mathcal{H}(\mathbf{S}, L, \text{MP}), \tag{8}$$
where $\mathcal{W}(\cdot)$ is total wirelength, the typical metric for floorplanning tasks [7], [11], [26], [27], and $\mathcal{H}(\cdot)$ captures latency cost [25], a metric specifically designed to address the challenges of 3D space by considering vertical stacking, die-to-die communication, and inter-layer connectivity. Θ includes all coordinates and die assignments, and μ balances layout compactness and inter-die efficiency.

After presenting the original objective function in Equation (8), we face the fundamental challenge of optimizing two heterogeneous terms, $\mathcal{W}(E)$ and $\mathcal{H}(\mathbf{S}, L, \text{MP})$, which cannot be directly unified. To resolve this, we decompose the second term and introduce weighted components. We express the objective as:
$$\mathcal{F}^o = (1-\mu)\cdot \sum_{i,j} A_{ij}^o \cdot d_{ij} + \mu \cdot \sum_{i,j} S_{ij}\cdot L_{ij}, \tag{9}$$
where A^o denotes the original connection matrix. To further unify these terms, we construct a weighted connection matrix:
$$A_{ij} = (1-\mu)\cdot A_{ij}^o + \mu \cdot S_{ij}\cdot \lambda_{ij}. \tag{10}$$
Thus, following the procedure in Section II-B, let $X \in \mathbb{R}^{n\times 3}$ represent the positions of the n modules in 3D space, with the definition of Equation (10), we can rewrite the objective as
$$\min_{\Theta} \quad \mathcal{F} = \sum_{i,j} A_{ij}\cdot d_{ij} = \langle A, D\rangle. \tag{11}$$
Finally, this unification converts the multi-objective optimization into a single weighted connection-distance matching problem and leverages the natural 3D structure of the 3D floorplanning domain.

III. METHODOLOGY OF GREAT3D

As shown in Fig. 2, GREAT3D adopts a two-stage framework. In contrast to prior methods that require netlist partitioning as a prerequisite, GREAT3D directly performs unified die assignment and block floorplanning within a native 3D solution space, enabling more globally optimal results. The first stage, detailed in Section III-A, solves the unified objective Equation (11) in the 3D space to obtain preliminary block placements, including their z-axis coordinates. These coordinates naturally guide the die assignment by splitting blocks between the top and bottom dies, providing inputs for the 2D LBFGS refinement described in Section III-B.

A. 3D SDP Optimization

Following the notations in Section II, the Gram matrix is defined as $\mathbf{G} = \mathbf{X}\mathbf{X}^\top$. Thus, the objective function is rewritten as Equation (12).
$$\langle \mathbf{A}, \mathbf{D}\rangle = \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{A}_{ij}\left(\mathbf{G}_{ii} + \mathbf{G}_{jj} - 2\mathbf{G}_{ij}\right). \tag{12}$$
Based on [26], we construct a matrix $\mathbf{B}$, to simplify the calculation of the matrix according to the mathematical transformation as Equation (13). This definition clearly distinguishes

the diagonal elements (which involve aggregated in- and out-connections of vertex i) from the off-diagonal ones (which are simply scaled by a constant). The $\mathbf{Z}$ is defined as Equation (14)

$$\mathbf{B}_{ij} = \begin{cases} \sum_{k=1}^{n} \mathbf{A}_{ik} + \sum_{k=1}^{n} \mathbf{A}_{kj}, & \text{if } i = j, \\ -2\,\mathbf{A}_{ij}, & \text{if } i \neq j. \end{cases} \tag{13}$$

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{X}_{3\times n}^{\top} \\ \boldsymbol{X}_{n\times3} & \boldsymbol{G}_{n\times n} \end{bmatrix} \in \mathbb{S}_{+}^{3+n}. \tag{14}$$

where $\boldsymbol{I}_{3\times3}$ is a 3×3 identity matrix that serves as the upper-left block of $\boldsymbol{Z}$, preserving the identity structure in this part of the matrix. Here, the construction needs a rank constraint, which is further explained in Equation (20). $\boldsymbol{G}_{n\times n}$ is a symmetric positive semidefinite matrix of dimensions $n\times n$, which forms the lower-right block of $\boldsymbol{Z}$ and captures additional relationships or dependencies among the n elements. The top-3 eigenvectors of the symmetric matrix $\mathbf{Z}$ are computed based on the Courant-Fischer principle, capturing the dominant spatial directions for recovering the 3D coordinates. Then the optimization is carried out by alternately solving two coupled subproblems. At each iteration, the updates of the solution variables are defined by the following abstract formulations shown in Equation (15) and Equation (16).

$$\mathbf{Z}^{(k+1)} = \mathcal{P}_1\big(\mathbf{Z}^{(k)}, \mathbf{W}^{(k)}; f_1, \mathcal{C}_1\big), \tag{15}$$

$$\mathbf{W}^{(k+1)} = \mathcal{P}_2\big(\mathbf{W}^{(k)}, \mathbf{Z}^{(k+1)}; f_2, \mathcal{C}_2\big), \tag{16}$$

$$f_1(\mathbf{Z}) = \langle \mathbf{B}, \mathbf{G}(\mathbf{Z})\rangle + \alpha\langle \mathbf{W}^*, \mathbf{Z}\rangle, \tag{17}$$

$$f_2(\mathbf{W}) = \langle \mathbf{W}, \mathbf{Z}^*\rangle. \tag{18}$$

where f_1 in Equation (17) and f_2 in Equation (18) denote the respective objective functions, and $\mathcal{C}_1$, $\mathcal{C}_2$ represent the associated constraint sets. The superscript k indicates the current iteration index, and $(k+1)$ denotes the updated solution in the subsequent step. After solving the SDP, we recover the 3D layout by performing spectral decomposition on the Gram matrix $\mathbf{G} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\top}$. The top-3 eigenvectors $\mathbf{V}3 \in \mathbb{R}^{n\times3}$, scaled by the square roots of their eigenvalues $\mathbf{\Sigma}\text{sqrt} \in \mathbb{R}^{3\times3}$, yield the recovered coordinates $\mathbf{X} = \mathbf{V}3\mathbf{\Sigma}\text{sqrt}$, each row of $\mathbf{X}$ corresponds to the spatial position of a module in 3D Euclidean space. The objective function of Subproblem 1 is defined as Equation (17), where $\mathbf{G}(\mathbf{Z})$ encodes the placement-induced distances. The second term is a regularization encouraging alignment with a fixed inter-die proximity matrix $\mathbf{W}^*$, controlled by parameter α. Subproblem 2 refines $\mathbf{W}$ by aligning it with the current layout solution $\mathbf{Z}^*$. To ensure non-overlapping placement, the pairwise distance matrix $\mathbf{Z}$ must satisfy Equation (19).

$$D_{ij} = \mathbf{Z}_{ii} + \mathbf{Z}_{jj} - 2\mathbf{Z}_{ij} \geq (r_i + r_j)^2, \quad \forall i \neq j, \quad \mathbf{Z} \succeq 0, \tag{19}$$

$$\mathbf{0} \preceq \mathbf{W} \preceq \mathbf{I}, \quad \text{Trace}(\mathbf{W}) = n \tag{20}$$

which ensures that any two blocks are separated by at least the sum of their radii and that $\mathbf{Z}$ remains positive semidefinite. In the dual formulation, $\mathbf{W}$ is constrained within a normalized semidefinite cone shown in Equation (20), where n is block number. These constraints ensure the eigenvalues of $\mathbf{W}$ lie between (0,1) and that the overall spectral weight is normalized.

Algorithm 1 2-D Refinement via L-BFGS-B

```
Input: Coordinates $\mathbf{X}_0$ from SDP in Section III-A
Output: Refined coordinates $\mathbf{X}^*$
 1: obj ← UPDATEOBJECTIVE();                  ▷ using Equation (21)
 2: $\mathbf{X} \leftarrow \mathbf{X}_0$;                              ▷ initialization
 3: while convergence criteria unmet do
 4:     $g$ ← UPDATEGRADIENT();               ▷ using Equation (24)
 5:     $\mathbf{p}$ ← UPDATEDIRECTION();              ▷ via Equation (27)
 6:     $\eta$ ← UPDATESTEP();                   ▷ using Equation (28)
 7:     $\mathbf{X} \leftarrow \mathbf{X} + \eta\,\mathbf{p}$;                     ▷ position update
 8:     UPDATEHISTORY();                      ▷ using Equation (29)
 9:     if UPDATETRIGGER($\mathbf{X}$) then
10:         UPDATEPARAMETERS();
11:     end if
12: end while
```

Finally, the SDP-based stage provides a globally informed 3D layout by minimizing a spectral relaxation of the unified objective (11), yielding block positions $\mathbf{X} \in \mathbb{R}^{n\times3}$.

B. 2D Refinement

Building on the global 3D solution in Section III-A, we further improve die layout quality through a second-stage refinement that operates within each die separately via L-BFGS-B procedure, which is shown in Algorithm 1.

We first construct the composite objective function $\texttt{obj} = f(\mathbf{X})$ using Equation (21), which integrates wirelength and spacing terms (line 1). Then the optimization is initialized with $\mathbf{X} \leftarrow \mathbf{X}_0$, where $\mathbf{X}_0$ is the projected 2D coordinate set from the SDP solution described in Section III-A (line 2). While convergence criteria are not met (line 3), we first compute the gradient $g = \nabla f(\mathbf{X})$ using Equations (24) and (25) (line 4).

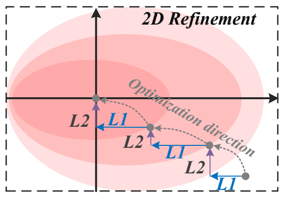
Next, a search direction $\mathbf{p}$ is then determined using the L-BFGS two-loop recursion [28], based on the current gradient g and memory history $\mathcal{M}$ [29] (line 5). To ensure sufficient descent, a step size η is computed via Armijo backtracking line search [30], satisfying the sufficient decrease condition Equation (28) (line 6). The block positions are then updated as $\mathbf{X} \leftarrow \mathbf{X} + \eta\,\mathbf{p}$ (line 7). After the update, the function UPDATEHISTORY() appends the latest (s_k, y_k) pair as defined in Equation (29) (line 8). If dynamic update criteria PARAMETERUPDATECONDITION($\mathbf{X}$) are satisfied (line 9), we adaptively update the penalty coefficient β and norm mixing ratio α_3 to improve convergence and maintain spacing robustness (line 10). The iteration continues until a stopping criterion is triggered, and the final intra-die placement $\mathbf{X}^*$ is returned (line 12).

Following the introduction of the framework of this stage, we then give an illustration of the details.

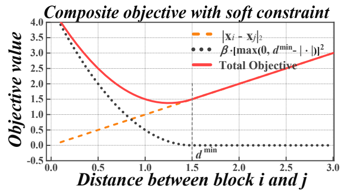
After the SDP in Section III-A, the objective function in this stage should composite wirelength objective augmented with pairwise soft-spacing constraints, so it is defined as Equation (21).

$$f(\mathbf{x}) = \sum_{i<j} \mathbf{A}_{ij} \cdot \phi(\mathbf{x}_i, \mathbf{x}_j) + \beta\, f_{\text{pen}}(\mathbf{x}). \tag{21}$$

$$\phi(\mathbf{x}_i, \mathbf{x}_j) = \alpha_3\|\mathbf{x}_i - \mathbf{x}_j\|_1 + (1-\alpha_3)\|\mathbf{x}_i - \mathbf{x}_j\|_2, \tag{22}$$



(a)



(b)

Fig. 3 Illustration of the 2D refinement strategy. (a) Alternating *L1* and *L2* gradients improve convergence. (b) Soft spacing penalty maintains minimum block separation requirements.

$$f_{\text{pen}}(\mathbf{x}) = \sum_{i<j} \left[\max\left(0,\ d_{ij}^{\min} - \|\mathbf{x}_i - \mathbf{x}_j\|_2\right)\right]^2. \tag{23}$$

where $\mathbf{x}_i = [x_i, y_i]^\top$ denotes the 2D position of block i acquired from the procedure in Section III-A for each die, and $\beta > 0$ is a regularization weight controlling the strength of spacing penalties. To enhance convergence and escape poor local minima, we adopt a norm-alternating strategy within the potential function Equation (22), where a scalar $\alpha_3 \in [0, 1]$ balances Manhattan and Euclidean terms. This composite form leverages the strengths of both norms, as previously shown effective in [26]. To maintain legal spacing between block pairs, we incorporate a soft penalty term f_{pen} defined as Equation (23), where $d_{ij}^{\min} > 0$ denotes the minimum spacing constraint between blocks i and j derived from geometric constraints in Section II-D. As illustrated in the right portion of Fig. 3(b), this quadratic penalty softly discourages overlaps without introducing hard constraints. The left portion of Fig. 3(a) visualizes our alternating-norm descent strategy, where L1 and L2 gradients are switched iteratively to improve search direction quality during optimization. With the objective function defined in Eq. (21), we compute gradients as Equation (24).

$$\frac{\partial f}{\partial \mathbf{x}_i} = \sum_{j\neq i} \mathbf{A}_{ij} \cdot \nabla_i \phi(\mathbf{x}_i, \mathbf{x}_j) - 2\beta \sum_{j\neq i} \Delta_{ij} \cdot \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|_2}, \tag{24}$$

$$\nabla_i \phi(\mathbf{x}_i, \mathbf{x}_j) = \alpha_3 \cdot \text{sign}(\mathbf{x}_i - \mathbf{x}_j) + (1-\alpha_3) \cdot \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|_2}, \tag{25}$$

$$\Delta_{ij} = \max\left(0,\ d_{ij}^{\min} - \|\mathbf{x}_i - \mathbf{x}_j\|_2\right). \tag{26}$$

Then, we define the update procedure shown in Equation (27), where H_k is the L-BFGS approximation of the inverse Hessian, computed using the two-loop recursion described in [29]. This procedure recursively updates the direction using a limited memory of past updates.

$$\mathbf{p} = -H_k \nabla f(\mathbf{X}_k), \tag{27}$$

Next, the Armijo backtracking line search [30] to compute a valid step size η is shown by Equation (28). The step is accepted only if it satisfies the sufficient decrease condition (line 6-7):

$$f(\mathbf{X} + \eta\,\mathbf{p}) \le f(\mathbf{X}) + c \cdot \eta \cdot \nabla f(\mathbf{X})^\top \mathbf{p}. \tag{28}$$

$$s_k = \mathbf{X}_k - \mathbf{X}_{k-1}, \quad y_k = \nabla f(\mathbf{X}_k) - \nabla f(\mathbf{X}_{k-1}). \tag{29}$$

where $c \in (0, 1)$ is a small constant (e.g., 10^{-4}). The optimizer then updates the block coordinates via $\mathbf{X} \leftarrow \mathbf{X} + \eta\,\mathbf{p}$. After each update, the function UPDATEHISTORY appends the new (s_k, y_k) pair. And we discards the oldest pair if the memory limit m is exceeded. These updates (line 8) enable L-BFGS to efficiently approximate second-order curvature without storing full Hessians. Finally, we monitor a set of dynamic refinement (line 9-11) via PARAMETERUPDATECONDITION($\mathbf{X}$), and adapt hyperparameters accordingly. Specifically, if any pairwise spacing violation exceeds a threshold, the penalty weight β is increased to enforce tighter spacing control. Similarly, the L1–L2 mix coefficient α_3 is adjusted periodically to enhance convergence. The optimization loop continues until convergence criteria are met, such as a small gradient norm or a maximum number of iterations. The refined layout $\mathbf{X}^*$ is returned as the output for Great3D framework, which is also the solution of the unified objective shown as Equation (11) for the 3D IC floorplanning problem proposed in Section II-D.

IV. EXPERIMENTAL RESULTS

A. Experimental Setup and Baseline Setup

All experiments are conducted using benchmarks from the GSRC benchmark suite [33], with the same of [26], [27]. The proposed 3D floorplanning algorithm is implemented in Python 3.11 and executed on a Linux workstation running Ubuntu 22.04 LTS, equipped with dual Intel Xeon Gold 6426Y processors and 256 GB RAM. The following 6 baseline methods mentioned in Section I are evaluated for comparison: (1) FM+AR, which combines the FM partitioning algorithm [13] with the AR 2D floorplanner [27]; (2) FM+Li, using FM partitioning and Li's 2D optimizer [26]; (3) Spectral+AR, using spectral partitioning [14] with AR [27]; (4) Spectral+Li, combining spectral partitioning with Li's floorplanner [26]; (5) GSP, a Grouped Sequence Pair-based 3D floorplanner [7]; and (6) TA3D, a thermal-aware 3D planner using smoothed HPWL and FM-style optimization [11].

B. Preformance Comparison

TABLE II compares GREAT3D with six baselines across five GSRC testcases, reporting both wirelength and runtime. Great3D already outperforms all state-of-the-art baselines in wirelength while keeping runtime competitive. On the

TABLE I Parameter Settings.

Parameter	Value	Ref.	Parameter	Value	Ref.
α_1	5×10^{-4}	[31]	α_2	1.8×10^{-1}	[31]
β_1	1.1×10^{-1}	[31]	MP	2.4×10^{-1}	[31]
CPI_{sta}	1.532×10^{-3}	[25]	β_2	9.8×10^{1}	[32]

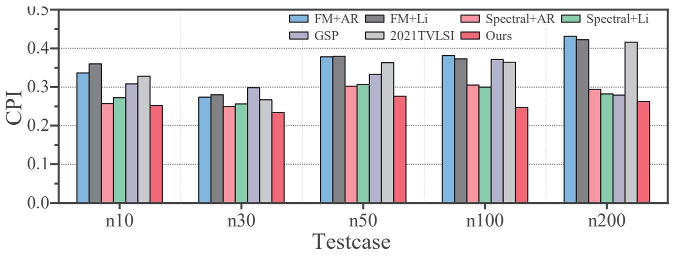


Fig. 4 Normalized Cycle-per-instruction (CPI), reporting $\mathcal{H}$ comparison on the GSRC block benchmarks.

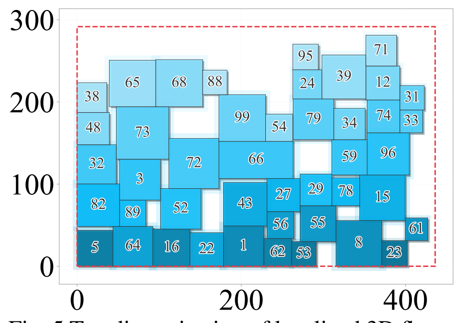


Fig. 5 Top die projection of legalized 3D floorplan with aspect ratio of 4:3. The red dashed outline represents the fixed die boundary constraint, with blocks arranged to minimize total wirelength.

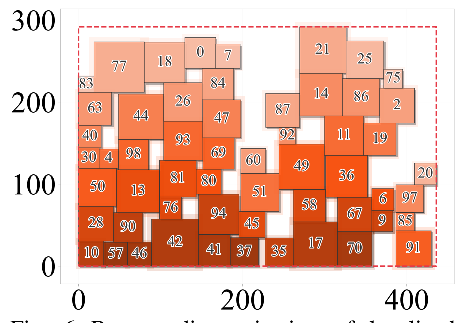


Fig. 6 Bottom die projection of legalized 3D floorplan with aspect ratio of 4:3. The red dashed outline represents the fixed die boundary constraint, with blocks arranged to minimize total wirelength.

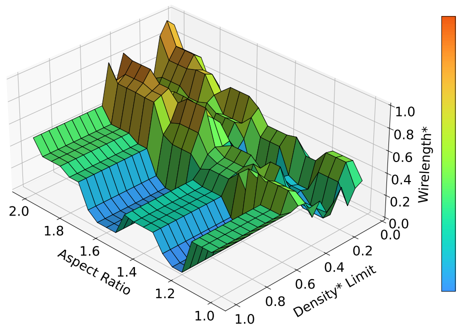


Fig. 7 Normalized wirelength surface under varying outline aspect ratio and inter-die connection constraint. The * represents the normalized value.

TABLE II Experimental results on GSRC benchmarks [33] with runtime values doubled. Wirelength is measured in μm and runtime in seconds.

Testcase	FM [12]+AR [27]		FM [12]+Li [26]		Spectral [14]+AR [27]		Spectral [14]+Li [26]		GSP [7]		TA3D [11]		Great3D (ours)	
	Wirelength	Time	Wirelength	Time	Wirelength	Time	Wirelength	Time	Wirelength	Time	Wirelength	Time	Wirelength	Time
n10	43223	6.72	48881	2.16	24571	5.04	28020	2.08	36444	7.62	41166	41.1	**23415**	17.12
n30	84456	13.84	88703	5.68	67193	15.68	72048	10.00	100818	41.86	79429	55.76	**56542**	25.2
n50	217731	21.44	219508	18.32	144133	32.56	147835	18.82	174368	109.74	203847	159.06	**119495**	42.88
n100	402324	37.04	388336	102.16	268341	45.36	259307	107.84	385398	351.9	373565	161.94	**166792**	91.92
n200	880486	130.16	852036	1074.32	446443	115.04	408192	1105.04	397594	1360.86	832867	772.84	**345099**	209.68
Avg. ratio	2.025	0.493	2.058	1.403	1.269	0.544	1.289	1.480	1.652	2.997	1.904	2.754	**1.000**	1.000

five GSRC benchmarks, its average normalised wirelength is 1.45×. In conclusion, while Baseline 1–4 may run slightly faster, all methods complete within acceptable runtime limits. Their early-stage partitioning severely handicaps their ability to optimize, resulting in significantly worse outcomes. Besides, Baseline 5 and 6 adopt stacking strategy along the z-axis, which leads to an exponential explosion in the 3D search space. This not only causes longer runtimes, often exceeding ours, but also fails to deliver better quality. Whereas, GREAT3D, maintains competitive runtime while consistently achieving significantly better results. This demonstrates that our method consistently achieves the best HPWL in Section II among all evaluated approaches.

Then we use Equation (1), following the parameters shown in TABLE I, to calculate the generalized CPIs as illustrated in the Fig. 4. This figure contrasts the CPI obtained by our GREAT3D framework with six state-of-the-art baselines. Because GREAT3D co-optimizes die assignment and block placement, it shortens the critical cross-die paths captured by L while simultaneously minimizing wirelength$_{ppi}$. As a result, our method lowers CPI by 8.4–15.7% across all benchmarks and delivers an average 11.2% improvement over the closest competitor (Spectral+Li, baseline 4), clearly demonstrating the performance benefit of a latency-aware native 3D flow, getting the best performance regarding the latency metric in Section II. To this end, the analytical Great3D has used the unified objective in the natural 3D optimization space, to get the best results in different metrics among different baselines.

As illustrated in Figs. 5 and 6, blocks are placed to optimize inter-die connectivity while respecting die boundaries (red dashed boxes). The complementary placement across dies balances area usage and supports wirelength optimization, with block IDs aiding cross-die reference and vertical connection analysis. Fig. 7 shows the normalized wirelength surface under varying outline aspect ratios and inter-die connection constraints. While theory suggests that elongating the outline increases wirelength, we observe non-monotonic trends due to the discrete nature of block configurations: some non-square ratios yield better layouts. For a fixed aspect ratio, reducing the inter-die constraint γ limits cross-die placement flexibility and often leads to suboptimal wirelength. Overall, the surface reveals that tighter inter-die limits degrade layout quality, while aspect ratio affects wirelength in a layout-sensitive manner.

V. CONCLUSION

This paper presented a native 3D floorplanning framework named GREAT3D, which combines partitioning, floorplanning, wirelength minimization, and system-level latency modeling within a unified analytical framework, eliminating separate partitioning stages. The experiments underscore the necessity of considering inter-die communication early in the design process, especially for 3D integration. Future research will focus on incorporating thermal management strategies into the proposed optimization framework, enabling better design.

ACKNOWLEDGMENT

The research work described in this paper was conducted in the JC STEM Lab of Intelligent Design Automation funded by The Hong Kong Jockey Club Charities Trust. This work is jointly supported by the Research Grants Council of Hong Kong SAR No. CUHK14211324 and No. CUHK7010840.

References

[1] K. Cao, J. Zhou, T. Wei, M. Chen, S. Hu, and K. Li, "A survey of optimization techniques for thermal-aware 3d processors," *Journal of Systems Architecture*, 2019.

[2] X. Zhao, W. Li, Z. Zeng, Z. Huang, B. Xie, X. Li, and Y. Bao, "Toward advancing 3D-ICs physical design: Challenges and opportunities," in *Proc. ASPDAC*, 2025.

[3] T. Widodo, X. Brun, P. Lianto, A. Tan, J. Lie, P. Lim, and G. See, "A cmp process for hybrid bonding application with conventional/nt-cu and si x n y/si x o y dielectrics," in *2024 IEEE 74th Electronic Components and Technology Conference (ECTC)*. IEEE, 2024, pp. 2058–2061.

[4] J. H. Lau, "State-of-the-art of cu-cu hybrid bonding," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 2024.

[5] S. Khushu and W. Gomes, "Lakefield: Hybrid cores in 3d package." in *Hot Chips Symposium*, 2019, pp. 1–20.

[6] L. Cheng, L. Deng, and M. D. Wong, "Floorplanning for 3-D VLSI design," in *Proc. ASPDAC*, 2005, pp. 405–411.

[7] R. K. Nain and M. Chrzanowska-Jeske, "Fast Placement-Aware 3-D Floorplanning Using Vertical Constraints on Sequence Pairs," *IEEE TVLSI*, 2011.

[8] S. Chen and T. Yoshimura, "Multi-layer floorplanning for stacked ICs: Configuration number and fixed-outline constraints," *Integration*, vol. 43, no. 4, pp. 378–388, 2010.

[9] R. Zhong, X. Du, S. Kai *et al.*, "FlexPlanner: Flexible 3D floorplanning via deep reinforcement learning in hybrid action space with multi-modality representation," *Proc. NeurIPS*, vol. 37, pp. 49 252–49 278, 2024.

[10] B. Fu, L. Liu, Y. Sun, W.-H. Lau, M. D. Wong, and E. F. Young, "Coplace: Coherent placement engine with layout-aware partitioning for 3D ICs," in *Proc. ASPDAC*, 2024, pp. 65–70.

[11] J.-M. Lin, W.-Y. Chang, H.-Y. Hsieh, Y.-T. Shyu, Y.-J. Chang, and J.-M. Lu, "Thermal-aware floorplanning and TSV-planning for mixed-type modules in a fixed-outline 3-D IC," *IEEE TVLSI*, vol. 29, no. 9, pp. 1652–1664, 2021.

[12] P. Vanna-Iampikul, C. Shao, Y.-C. Lu *et al.*, "Snap-3D: A constrained placement-driven physical design methodology for high performance 3-D ICs," *IEEE TCAD*, vol. 42, no. 7, pp. 2331–2335, 2022.

[13] L.-T. Liu, M.-T. Kuo, S.-C. Huang, and C.-K. Cheng, "A gradient method on the initial partition of fiduccia-mattheyses algorithm," in *Proc. ICCAD*, 1995, pp. 229–234.

[14] C. J. Alpert and S.-Z. Yao, "Spectral partitioning: The more eigenvectors, the better," in *Proc. DAC*, 1995, pp. 195–200.

[15] E. Beyne, "The 3-d interconnect technology landscape," *IEEE Design & Test*, 2016.

[16] Y. Zhao, L. Zou, and B. Yu, "Invited: Physical design for advanced 3D ICs: Challenges and solutions," in *Proc. ISPD*, 2025.

[17] J. Kim, L. Zhu, H. M. Torun, M. Swaminathan, and S. K. Lim, "Micro-bumping, hybrid bonding, or monolithic? a ppa study for heterogeneous 3d ic options," in *Proc. DAC*. IEEE, 2021, pp. 1189–1194.

[18] T. Lu, C. Serafy, Z. Yang, S. K. Samal, S. K. Lim, and A. Srivastava, "TSV-based 3-D ICs: Design methods and tools," *IEEE TCAD*, vol. 36, no. 10, pp. 1593–1619, 2017.

[19] S. S. K. Pentapati and S. K. Lim, "Heterogeneous Monolithic 3D ICs: EDA Solutions, and Power, Performance, Cost Tradeoffs," in *Proc. DAC*, 2021, pp. 925–930.

[20] K. Ma, N. Bekiaris, S. Ramaswami, T. Ding, G. Probst, J. Burggraf, and T. Uhrmann, "0.5 $1/4$m pitch wafer-to-wafer hybrid bonding with SiCN bonding interface for advanced memory," in *Proc. ECTC*, 2023, pp. 1110–1114.

[21] C. Netzband, K. Ryan, Y. Mimura, S. Ilseok, H. Aizawa, N. Ip, X. Chen, H. Fukushima, and S. Tan, "0.5 μm pitch next generation hybrid bonding with high alignment accuracy for 3d integration," in *Proc. ECTC*, 2023.

[22] S. H. Hahn, W. Kim, D. Shin, Y. Lee, S. Kim, W. Choi, K. Lim, B. Moon, and M. Rhee, "Contamination-free Cu/SiCN hybrid bonding process development for sub-μm pitch devices with enhanced bonding characteristics," in *Proc. ECTC*, 2023, pp. 1390–1396.

[23] S. Liu, J. Jiang, Z. He, Z. Wang, Y. Lin, B. Yu, and M. Wong, "Routing-aware legal hybrid bonding terminal assignment for 3d face-to-face stacked ics," in *Proceedings of the 2024 International Symposium on Physical Design*, 2024.

[24] A. Zhou, Y. Zhang, F. Ding, Z. Lian, R. Jin, Y. Yang, Q. Wang, and L. Cao, "Research progress of hybrid bonding technology for three-dimensional integration," *Microelectronics Reliability*, vol. 155, p. 115372, 2024.

[25] Z. Zhuang, K.-Y. Chao, B. Yu, T.-Y. Ho, and M. D. Wong, "Multi-product optimization for 3D heterogeneous integration with D2W bonding," in *Proc. ICCAD*, 2023.

[26] W. Li, F. Wang, J. M. F. Moura, and R. D. Blanton, "Global floorplanning via semidefinite programming," in *Proc. DAC*, 2023.

[27] C. Luo, M. F. Anjos, and A. Vannelli, "Large-scale fixed-outline floorplanning design using convex optimization techniques," in *2008 Asia and South Pacific Design Automation Conference*. IEEE, 2008, pp. 198–203.

[28] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on mathematical software (TOMS)*, vol. 23, no. 4, pp. 550–560, 1997.

[29] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on scientific computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[30] J. Nocedal and S. J. Wright, "Nonlinear equations," *Numerical Optimization*, pp. 270–302, 2006.

[31] R. Clapp, M. Dimitrov, K. Kumar, V. Viswanathan, and T. Willhalm, "Quantifying the performance impact of memory latency and bandwidth for big data workloads," in *2015 IEEE International Symposium on Workload Characterization*. IEEE, 2015, pp. 213–224.

[32] Y. Zhang, X. Hu, A. Deutsch, A. E. Engin, J. F. Buckwalter, and C.-K. Cheng, "Prediction and comparison of high-performance on-chip global interconnection," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 19, no. 7, pp. 1154–1166, 2010.

[33] GSRC Benchmark Suite, "GSRC Floorplanning Benchmark Suite," http://vlsicad.eecs.umich.edu/BK/GSRCbench/, accessed: September 15, 2024.