Integrated Timing-driven Placement for Hybrid-Bonding-based Face-to-Face 3D ICs

Yuhao Ji[1,3], Yunqi Shi[5,6], Tianshu Hou[1], Yuxuan Zhao[1], Chunyuan Zhao[2,3], Peiyu Liao[1], Zizheng Guo[2,3], Chao Qian[5,6], Yibo Lin[2,3,4], Bei Yu[1]
[1]Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong SAR
[2]School of Integrated Circuits, Peking University, Beijing, China
[3]Institute of Electronic Design Automation, Peking University, Wuxi, China
[4]Advanced Innovation Center for Integrated Circuits, Beijing, China
[5]State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, China
[6]School of Artificial Intelligence, Nanjing University, Nanjing, China

Abstract

With the deceleration of Moore's Law, three-dimensional (3D) integration via hybrid bonding enables higher performance. However, existing 3D placers exhibit limitations: pseudo-3D placers suffer from modeling inaccuracies, while true-3D placers neglect timing closure. Prior 3D static timing analysis (STA) lacks accurate parasitic modeling, impeding timing optimization. This paper presents the first timing-driven true-3D placement framework for hybrid-bonding-based integrated circuits. We develop a 3D STA engine with accurate parasitic modeling and integrate a sample-based 3D wirelength model with hybrid weighting for global placement, followed by Lagrangian-based detailed placement. Results demonstrate 30 ~ 1002× reduction in TNS and 3 ~ 14× reduction in WNS over state-of-the-art placers.

1 Introduction

As Moore's Law scaling faces mounting fabrication challenges and rising costs, the semiconductor industry has shifted toward three-dimensional (3D) integration to achieve higher functional density, improved performance, and reduced power requirements [1]. As illustrated in Figure 1(a), wafer-to-wafer (W2W) hybrid bonding has emerged as a leading 3D integration approach, enabling face-to-face (F2F) inter-die interconnections through direct Cu-Cu bonding with fine pitch, negligible parasitics, and therefore high bandwidth and low energy consumption [2, 3]. Due to these benefits, it has been widely adopted in high-performance computing, both in industry [2, 4] and academia [5–7].

While hybrid bonding offers compelling advantages, it introduces unique physical design challenges that demand advanced electronic design automation (EDA) tools. Placement is a critical stage in the physical design flow. Figure 1(b) illustrates a mixed-size 3D placement problem, which determines instance-to-die assignments while optimizing instance locations within the layout space. Existing 3D placers can be broadly categorized as pseudo-3D or true-3D approaches. **Pseudo-3D placement** leverages commercial 2D tools by constructing emulated layouts that approximate 3D designs. For example,

This work is supported by The Research Grants Council of Hong Kong SAR (No. CUHK14211824 and No. CUHK14201624).

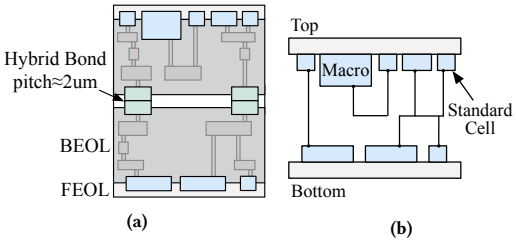

Figure 1: (a) Hybrid-bonding-based F2F 3D integration. (b) Mixed-size 3D placement problem for F2F 3D ICs.

Shrunk-2D [8] constructs a 2D layout that emulates the final 3D IC footprint by shrinking cells and interconnects by 50%, followed by grid-wise min-cut partitioning to resolve overlaps. Compact-2D [9] employs a layout compaction strategy that scales RC parasitics in a pseudo-3D layout before performing tier partitioning. While benefiting from mature 2D tool infrastructure, these placers inherently suffer from modeling inaccuracies when approximating 3D geometries and parasitic effects in 2D space. Moreover, complex heuristics for macro partitioning are typically required, which limits their generalizability to diverse design scenarios.

In contrast, recent analytical **true-3D placement** frameworks [10–12] have demonstrated superior wirelength optimization by directly modeling 3D objectives on academic benchmarks [13, 14]. However, these works also exhibit limitations. First, they focus exclusively on wirelength minimization while neglecting power-performance-area (PPA) optimization, particularly timing closure. Previous 2D placement studies [15, 16] have highlighted the substantial gap between wirelength and actual timing performance, and therefore timing-driven optimization remains largely unexplored in 3D contexts. Second, existing benchmarks [13, 14] lack realistic I/O pin configurations, limiting practical applicability. Other works that incorporate timing considerations, such as TA-3D [17] and Open3DBench [18], adopt netlist-level partitioning followed by 2.5D placement rather than holistic 3D optimization in the complete solution space. These limitations underscore the critical need for timing-driven true-3D placement methodologies.

However, enabling timing-driven 3D placement requires accurate and efficient static timing analysis (STA) to guide optimization through millions of incremental design transformations [19–21]. Existing 3D STA engines that adapt commercial 2D tools through workarounds suffer from modeling deficiencies. Specifically, they lack native RC network modeling for 3D nets, and 3D interconnections are either modeled as zero-capacitance routing vias [22], dummy buffer

DAC '26, Long Beach, CA, USA

ACM ISBN 979-8-4007-2254-7/2026/07
https://doi.org/10.1145/3770743.3804060

cells with manually annotated net delays [23], or represented through separate SPEF files for cross-die parasitics [9]. These workarounds compromise timing accuracy and impose substantial manual overhead, rendering them impractical for scalable timing optimization.

To address these limitations, we present the first timing-driven true-3D placement framework for F2F hybrid-bonding-based 3D integrated circuits (ICs). Our approach is enabled by a native 3D STA engine that accurately models hybrid bond parasitics and provides guidance during placement optimization. We evaluate our framework on the open-source 3D design flow Open3DBench [18] with I/O pin configurations, demonstrating significant timing improvements over prior 3D placers. The main contributions of this paper are as follows:

- We develop a native 3D STA engine, explicitly modeling hybrid bond parasitics in the RC network of each 3D net and supporting fast incremental timing updates for placement optimization.
- We propose a timing-driven 3D global placement algorithm with a sample-based 3D wirelength model for smooth partition optimization and a hybrid net-pin weighting strategy.
- We propose a timing-driven 3D detailed placement algorithm to solve the Lagrangian dual problem of 3D timing optimization.
- We introduce a timing-driven 2.5D global placement stage to co-optimize hybrid bonding terminal (HBT) and instance positions.

2 Problem Formulation

This paper addresses the timing-driven mixed-size placement problem for hybrid-bonding-based F2F 3D ICs. Given a design netlist with standard cells and macros, cell libraries, timing constraints, HBT specifications, I/O pin locations and layout constraints, including die dimensions and row configurations, the objective is to partition and then place all the instances on the top or bottom die for better timing performance while satisfying design constraints. Following the ICCAD 2023 contest [14], we enforce the following design constraints:

- All the instances must be non-overlapping. The standard cells are aligned to rows and sites. HBT spacing constraints are satisfied.
- Each instance is assigned to either top or bottom die, and the maximum utilization of each die must be satisfied.

Our framework incorporates realistic I/O pin configurations and performs global routing using Open3DBench [18]. This enables timing evaluation with accurate parasitic extraction for both routing wires and hybrid bonds. The final evaluation metrics include TNS, WNS and overall runtime. We currently assume homogeneous technology nodes for both dies due to the evaluation flow's limitation, but our methodology extends to heterogeneous scenarios.

3 The Proposed Framework

The overall framework is illustrated in Figure 2. It begins with a 3D placement stage, which optimizes the instance partitioning and locations simultaneously in a complete 3D solution space. Subsequently, the framework introduces a 2.5D placement stage, which refines the layout and derives the legalized placement result.

3.1 3D Static Timing Analysis

Timing driven placement requires accurate and efficient static timing analysis (STA) to guide iterative optimization [15, 16, 24]. In this work, we develop a native 3D STA engine that explicitly models each hybrid bond as a wire segment with parasitic resistance and capacitance in the RC network.

In the 3D placement stage, HBT locations have not yet been determined. To enable timing-aware optimization at this stage, we adopt a virtual HBT approach following the ICCAD benchmark assumption [13, 14], where each inter-die net is connected with a single HBT. For each inter-die net e, we determine the virtual HBT location using a wirelength-minimization strategy [10]. Let $\mathcal{B}^+ = [x^+_{\min}, x^+_{\max}] \times [y^+_{\min}, y^+_{\max}]$ and $\mathcal{B}^- = [x^-_{\min}, x^-_{\max}] \times [y^-_{\min}, y^-_{\max}]$ denote the bounding boxes of pins on the top and bottom dies, respectively. The virtual HBT is placed at the center of the overlap region $\mathcal{B}^* = [x^*_{\min}, x^*_{\max}] \times [y^*_{\min}, y^*_{\max}]$,

$$(x_{\text{HBT}}, y_{\text{HBT}}) = \left((x^*_{\min} + x^*_{\max})/2,\ (y^*_{\min} + y^*_{\max})/2\right), \tag{1}$$

where $x^*_{\min}$ and $x^*_{\max}$ are the two median values of $\{x^+_{\min}, x^-_{\min}, x^+_{\max}, x^-_{\max}\}$, and $y^*_{\min}$, $y^*_{\max}$ are defined similarly.

With the virtual HBT serving as a shared terminal, each 3D net is decomposed into two 2D nets, one on the top die and one on the bottom die. This decomposition is well-justified by recent work on native F2F 3D routing [25], where 3D nets are decomposed into 2D subnets and routed independently on each die. Subsequently, for a net with n pins, where a pins are located on the top die and the remaining $(n-a)$ pins on the bottom die, we construct routing trees on each die using FLUTE [26] to construct rectilinear Steiner minimum trees (RSMTs), treating the virtual HBT location as an additional terminal. This yields RSMTs with $2a$ and $2(n-a)$ points on the top and bottom dies, respectively. The two RSMTs are connected through a hybrid bond edge with fixed parasitic parameters, forming a composite 3D routing tree as illustrated in Figure 3. We then extract RC parasitics from this structure and compute net delays using the Elmore delay model [27], propagating slews and arrival times through the timing graph for global timing analysis.

Furthermore, to support efficient timing analysis during iterative optimization, our STA engine provides an incremental update mechanism by caching routing tree topologies. This capability is particularly elaborated in our timing-driven 3D detailed placement in Section 3.3. Notably, our STA engine is flexible and supports both virtual and physical HBT modeling. In the 2.5D refinement stage discussed in Section 3.4, after HBT insertion, the engine allows switching to using exact HBT locations for more accurate timing analysis, thus enabling co-optimization of instances and HBTs. Our 3D STA engine is built upon HeteroSTA [28], with its binary available at https://heterosta3d.pkueda.org.cn/.

3.2 Timing-driven 3D Global Placement

In this section, we develop a timing-driven 3D global placer with optimized wirelength model and weighting strategies. The 3D global placement stage optimizes the positions of all movable blocks, including standard cells and macros, within a 3D space. Following prior works [10–12], the placement region is defined as $w = [0, x_h] \times [0, y_h] \times [0, z_h]$, where the bottom die occupies $w^- = [0, x_h] \times [0, y_h] \times [0, z_h/2)$ and the top die occupies $w^+ = [0, x_h] \times [0, y_h] \times [z_h/2, z_h]$. All blocks are modeled with uniform depth $z_h/2$, and each block i has center coordinates (x_i, y_i, z_i) with $z_i \in [z_h/4, 3z_h/4]$ ensuring the blocks inside the region. Blocks are assigned to the bottom die when $z_i < z_h/2$ and the top die when $z_i \geq z_h/2$. Following the analytical placement paradigm [11, 16, 24, 30, 31], we solve the 3D global

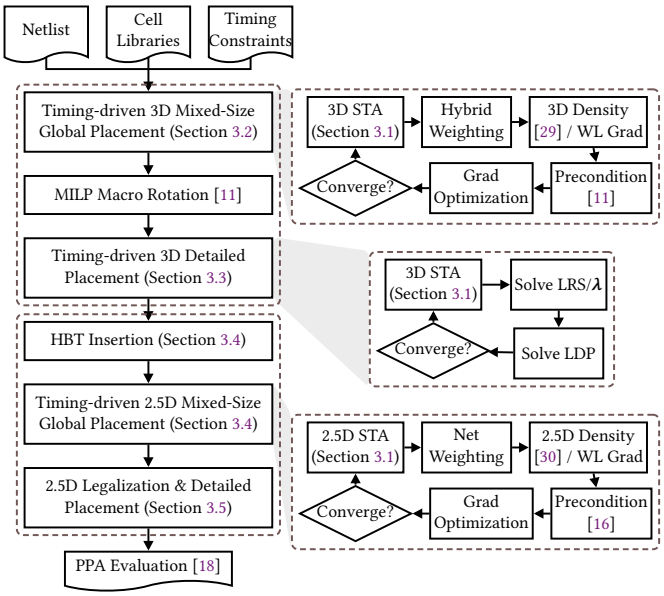

Figure 2: The proposed timing-driven F2F 3D placement flow.

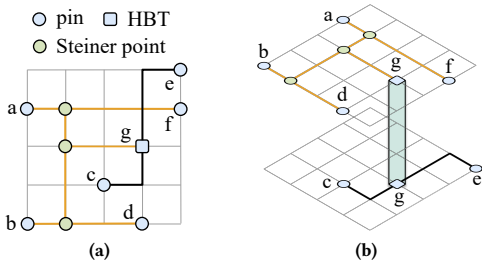

Figure 3: Illustration of 3D net routing tree for RC parasitics extraction. g represents the HBT. {a, b, d, f, g} belongs to the top die net e^+ and {c, e, g} belongs to the bottom die net e^-.

placement using gradient-based optimization with the objective:

$$\min_{\boldsymbol{p}} \sum_{e\in E} (\mathrm{WL}_e(\boldsymbol{p}) + N_e(\boldsymbol{p})) + \lambda \cdot D(\boldsymbol{p}), \tag{2}$$

where $\boldsymbol{p} = (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ represents the physical coordinates of the blocks and WL_e, N_e, and D represent the 3D wirelength, HBT cost and density penalty, respectively. The 3D density is modeled by eDensity3D model [29], which computes the potential map by solving the 3D Poisson's equation. Although the pitch of HBTs is fine, enabling high-density connectivity, some recent designs have utilized HBTs for power delivery network (PDN) in addition to signal routing [2]. Therefore, it is prudent to constrain the number of signal HBTs, and we incorporate the penalty term $N_e = \alpha \cdot p_e(\boldsymbol{z})$, where α is a weight factor. We define $p_e(\cdot)$ as the bounding box span of the net e along the corresponding axis, and adopt the weighted-average (WA) model [32] to obtain a differentiable relaxation:

$$p_e(\boldsymbol{x}) = \sum_{v\in e} x_v e^{\frac{1}{\tau_m}x_v} / \sum_{v\in e} e^{\frac{1}{\tau_m}x_v} - \sum_{v\in e} x_v e^{-\frac{1}{\tau_m}x_v} / \sum_{v\in e} e^{-\frac{1}{\tau_m}x_v}, \tag{3}$$

where τ_m is a parameter governing the smoothness.

Sample-based 3D Wirelength Model. The bistratal wirelength model proves to be effective for 3D placement since it better matches the die-to-die wirelength compared to naive 3D Half-Perimeter Wirelength (HPWL) model [10]. Given the pin positions $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$, which are calculated based on block positions and pin offsets, the bistratal

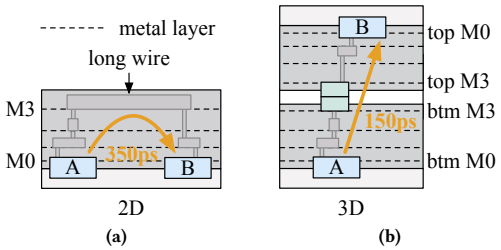

Figure 4: (a) 2D and (b) 3D routing of a long wire, showing the x-y plane wirelength can be reduced by cross-die placement.

wirelength model is defined as follows:

$$\begin{aligned}\mathrm{WL}_e^{\mathrm{bi}}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = &\max\{p_e(\boldsymbol{x}), p_{e^+}(\boldsymbol{x}) + p_{e^-}(\boldsymbol{x})\} + \\ &\max\{p_e(\boldsymbol{y}), p_{e^+}(\boldsymbol{y}) + p_{e^-}(\boldsymbol{y})\},\end{aligned} \tag{4}$$

where $+/-$ superscripts indicate the top or bottom nets, respectively, determined by the pin location $\boldsymbol{z}$. However, the bistratal wirelength model is discontinuous with respect to $\boldsymbol{z}$. When a pin crosses the die boundary at $z_h/2$, the die assignment changes abruptly, causing a discontinuity in the wirelength function and thus limiting the applicability of gradient-based optimization. Prior work [10] employs finite-difference approximation (FDA) to approximate gradients. In this work, to avoid discretization entirely and enable smoother optimization, we introduce a sample-based wirelength model, thereby casting the $\boldsymbol{z}$-variable optimization into a differentiable framework.

Instead of using the hard assignment rule $d_v = \mathbb{I}[z_v \geq z_h/2]$, where $d_v = 1$ indicates that the pin v is located on the top die, we use $\pi_k(z)$ to denote the assignment probability for a pin with location z and define $\pi_1(z) = \sigma((z - z_h/2)/\tau_s)$, $\pi_0(z) = 1 - \pi_1(z)$. Accordingly, for each pin $v \in e$ located at z_v, the assignment variable is distributed as $d_v|z_v \sim \mathrm{Bernoulli}(\pi_1(z_v))$. We define:

$$\mathrm{WL}_e^{\mathrm{samp}}(z_v) = \mathrm{WL}_e^{\mathrm{bi}}(\boldsymbol{x}, \boldsymbol{y}; d_v), \tag{5}$$

following the stochastic sampling scheme. Let $g_{v0}, g_{v1} \sim \mathrm{Gumbel}(0, 1)$ [33]. Since sampling is still non-differentiable, we then introduce a binary variable $\hat{d}_v$ via Gumbel-Max trick:

$$\hat{d}_v = \underset{k\in\{0,1\}}{\mathrm{argmax}} \ \log \pi_k(z_v) + g_{vk}, \tag{6}$$

which follows the same distribution as d_v. The exact sampling is achieved by adding independent Gumbel noise g_{vk} to the log-probabilities $\log \pi_k(z_v)$, and selecting the largest element [33]. Finally, to achieve a differentiable relaxation for argmax, we use the Softmax with temperature $\tau_o > 0$:

$$\mathrm{score}_{vk}(z_v) = \exp\{\frac{\log \pi_k(z_v) + g_{vk}}{\tau_o}\} / \sum_{k'\in\{0,1\}} \exp\{\frac{\log \pi_{k'}(z_v) + g_{vk'}}{\tau_o}\}. \tag{7}$$

By substituting Equation (7) for the discrete die assignment d_v in Equation (5), the wirelength takes the form :

$$\mathrm{WL}_e^{\mathrm{samp}}(z_v) = \sum_{k\in\{0,1\}} \mathrm{score}_{vk} \cdot \mathrm{WL}_e^{\mathrm{bi}}(\boldsymbol{x}, \boldsymbol{y}; k), \tag{8}$$

yielding a smooth and differentiable modeling with respect to z_v by considering the possible die assignments weighted by their scores. Furthermore, we introduce a quadratic pin-to-source wirelength model to avoid extremely long wires, which can significantly degrade the downstream timing performance:

$$\mathrm{WL}_e^{\mathrm{quad}} = \sum_{v\in e} \mathrm{dist}_{xy}(v, s(e))^2, \tag{9}$$

where $\text{dist}_{xy}(v, s(e))^2 = (x_v - x_{s(e)})^2 + (y_v - y_{s(e)})^2$ is the squared x-y plane Euclidean distance between pin v and source $s(e)$. The overall wirelength model with weight parameter β is then formulated as:

$$\text{WL}_e = \text{WL}_e^{\text{samp}} + \beta \cdot \text{WL}_e^{\text{quad}}. \tag{10}$$

Hybrid Net-Pin Weighting Strategy. Previous works [15, 16, 34] have proposed a series of weighting strategies to guide the timing-driven placement optimization in 2D designs, demonstrating the effectiveness of net or pin weighting. In this work, we propose a hybrid net-pin weighting strategy for our 3D global placer.

Our observation is that for long intra-die wires in conventional 2D designs, routers tend to elevate them to the top metal layer via vertical vias, as illustrated in Figure 4(a), since top metal layers exhibit low parasitic resistance and capacitance. However, these wires eventually descend back to the bottom metal layer to connect to cells at the FEOL layer, traversing the entire metal stack both upward and downward. Similarly, inter-die wires in 3D nets with F2F bonding must also traverse the entire metal stack on both top and bottom dies, since cells are placed at the FEOL layer of each die. The crucial difference is that the 3D scenario offers a unique opportunity: by placing timing-critical pins on different dies, as illustrated in Figure 4(b), it is possible to shorten the wirelength in the xy-plane and thereby reduce routing delay. This is particularly advantageous given that hybrid bonds have relatively small parasitics and are not the primary contributors to delay for long wires. Therefore, we encourage placing timing-critical pins that are spatially distant from the source pin on a different die from the source pin. Given a net e with source pin $s(e)$, for each non-source (sink) pin v, the z-axis distance $\text{dist}_z(v, s(e))$ enters Equation (2), weighted by pin weight w_v from a custom criticality:

$$\begin{aligned} \text{criticality}_v &= \sigma \cdot \text{dist}_{xy}^{\text{norm}}(v, s(e)) + \text{slack}_v/\text{WNS}, \\ w_v &= \exp(-\tau \cdot \text{criticality}_v), \end{aligned} \tag{11}$$

where σ is a hyperparameter, and $\text{dist}_{xy}^{\text{norm}}(v, s(e))$ denotes the x-y plane distance normalized by the maximum distance. By introducing w_v with $\tau > 0$, we relax the z-axis constraints between source and timing-critical sinks with long source-sink distance.

Furthermore, previous 2D works [16, 24] suggest that timing closure can be achieved by reducing the wirelength of timing-critical nets. To this end, we also adopt a net-weighting strategy on the wirelength term. Let $\text{slack}_e = \text{slack}_{s(e)}$, since the source pin is the most critical pin in the net and reflects the criticality. We define the net weight $w_e = \text{slack}_e/\text{WNS}$. To prevent abrupt weight changes that may destabilize the optimization process, we incorporate a momentum-based update mechanism following [16] for both pin and net weights. Let $\bar{w}_e$ and $\bar{w}_v$ denote the momentum-smoothed weights. The detailed update formulas can be found in [16] and are omitted in this paper. Finally, the objective defined in Equation (2) is reformulated by integrating the weighting strategies:

$$\min_{\boldsymbol{p}} \sum_{e \in E} \left(\bar{w}_e \cdot \text{WL}_e + \sum_{v \in e} \bar{w}_v \cdot \text{dist}_z(v, s(e)) + N_e \right) + \lambda \cdot D. \tag{12}$$

3.3 Timing-driven 3D Detailed Placement

After global placement, we obtain an initial placement solution with partition results, followed by a MILP-based macro rotation [11] to optimize the wirelength. Detailed placement is then introduced for incremental optimization. Conventional 2D detailed placement refines the placement result by making discrete local adjustments [16, 35].

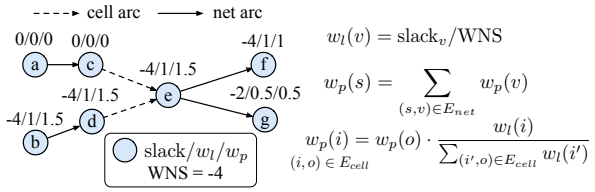

Figure 5: Timing weight propagation in 3D detailed placement.. For explicitness, w_p is not normalized by the maximum yet.

In this work, we extend the search space and propose a 3D timing-driven detailed placement algorithm. The pseudo code is detailed in Algorithm 1.

Consider a timing graph $G = (V, E)$, where V is the set of pins and E is the set of timing arcs. Let at_v denote the arrival time at pin v and delay_{ij} denote the timing arc delay from pin i to pin j. We formulate the timing optimization problem with TNS as the objective:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{d}} \quad & -\sum_{v \in \text{POs}} \text{slack}_v, \\ \text{s.t.} \quad & \text{at}_i + \text{delay}_{ij} \le \text{at}_j, \ \text{delay}_{ij} \ge 0, \quad (i,j) \in E, \\ & \text{slack}_v \le 0, \ \text{slack}_v \le T - \text{at}_v, \quad v \in \text{POs}, \end{aligned} \tag{13}$$

where T is the clock period, $\boldsymbol{d}$ is the partition results for all instances.

Notably, this formulation decouples optimization from delay computation, allowing arbitrary timing models for net and cell arcs in both 2D and 3D scenarios. We then derive the dual problem of Equation (13) using Lagrangian relaxation:

$$\begin{aligned} \max_{\boldsymbol{\lambda} \ge 0} \min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{d}} L_\lambda(\boldsymbol{x},\boldsymbol{y},\boldsymbol{d}) = & -\sum_{v \in \text{POs}} \text{slack}_v + \sum_{v \in \text{POs}} \lambda_v^0 \text{slack}_v \\ & + \sum_{(i,j) \in E} \lambda_{ij}(\text{at}_i + \text{delay}_{ij} - \text{at}_j) + \sum_{v \in \text{POs}} \lambda_v^1(\text{slack}_v + \text{at}_v - T), \end{aligned} \tag{14}$$

$$\begin{aligned} \text{s.t.} \quad & \lambda_v^0 + \lambda_v^1 = 1, \ \lambda_v^1 = \sum_{(i,v) \in E} \lambda_{iv}, \quad v \in \text{POs}, \\ & \sum_{(i,v) \in E} \lambda_{iv} = \sum_{(v,i) \in E} \lambda_{vi}, \quad v \in V/\text{POs}. \end{aligned} \tag{15}$$

Upon applying the flow constraints in Equation (15) to the Lagrangian function $L_\lambda(\boldsymbol{x},\boldsymbol{y},\boldsymbol{d})$, and approximately solving only the net delay part cell-by-cell, we get the simplified subproblem sLRS/$\boldsymbol{\lambda}$ [16]:

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{d}} \sum_{(i,j) \in E} \lambda_{ij} \text{delay}_{ij} \approx \min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{d}} \sum_{(i,j) \in E_{\text{net}}(c)} \lambda_{ij} \text{delay}_{ij}, \forall \text{ cell } c, \tag{16}$$

where $E_{\text{net}}(c)$ is the set of the connected net arcs of cell c. The Lagrangian dual problem (LDP) is thus formulated as follows:

$$\max_{\boldsymbol{\lambda} \ge 0} L_\lambda^* = \min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{d}} \sum_{(i,j) \in E} \lambda_{ij} \cdot \text{delay}_{ij} - T \sum_{v \in \text{POs}} \lambda_v^1, \tag{17}$$

with the constraints in Equation (15).

Solving LDP. According to Equation (16), the Lagrangian multipliers λ_{ij} serve as arc weights in the timing-driven optimization. We define two complementary weights for each pin v: local weight $w_l(v) = \text{slack}_v/\text{WNS}$ which is the normalized measurement of timing violation, and propagated weight $w_p(v)$ quantifying the global impact on endpoint violations, which satisfies the flow conservation constraint in Equation (15). As illustrated in Figure 5, $w_p(v)$ propagates through the timing graph in reverse topological order. Starting from primary outputs (POs) where $w_p = w_l$, each driver pin aggregates w_p from its fanout sink pins. These aggregated weights are then distributed to the cell input pins proportionally according to their local weights. Finally, w_p values are normalized by the maximum.

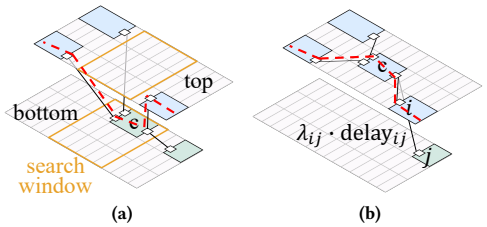

Figure 6: (a) Before (b) After 3D local discrete search for cell c. The yellow box is the search window. The critical path (red dash line) delay is reduced after movement.

For each net arc $(i, j) \in E_{net}$, we define $\lambda_{ij} = \lambda_i = w_l(i) + \beta \cdot w_p(i)$ using the source pin i's weights, since modifying any net arc affects all downstream delays within the net under existing delay models. By penalizing a net arc according to the weight of the source pin, we prevent scenarios where optimizing a non-critical sink inadvertently degrades critical paths sharing the same driver.

Solving sLRS/λ. To enable parallel cell movement without conflicts, we construct a maximal independent set $\mathcal{S}$ using a greedy algorithm. Cells are first sorted in descending order by the λ values of their output pins. Starting from the most critical cell, we iteratively select cells into $\mathcal{S}$ that do not share nets with any previously selected cells. This is achieved by maintaining a marked set $\mathcal{M}$ that includes both selected cells and their neighbors, i.e. cells sharing common nets. A cell is added to $\mathcal{S}$ only if it is not in $\mathcal{M}$, guaranteeing that cells in $\mathcal{S}$ can be optimized simultaneously without net delay conflicts. For each cell c in $\mathcal{S}$, we define a 3D search window consisting of two rectangular regions on the top and bottom dies with identical x-y plane projections. As illustrated in Figure 6, cell c is allowed to be reassigned to another die while exploring lateral positions.

To efficiently evaluate net delays across candidate positions, our 3D STA engine supports maintaining cached routing tree topologies for incremental timing analysis. For each Steiner point, we record the terminals that determine its coordinates during routing tree construction. When a cell moves to a candidate position, we incrementally update only affected nets by adjusting the location of the moved terminals and propagating coordinate changes to dependent Steiner points, while preserving the routing tree topology. This strategy provides two key benefits. First, it avoids time-consuming RSMT reconstruction for improved efficiency while maintaining accuracy within a local search window. Second, it guarantees optimization continuity by preserving topology, since complete reconstruction causes abrupt structural changes that lead to discontinuities and local optima, even in discrete local search. For each candidate position, we compute the weighted total cost $\sum_{(i,j)\in E_{net}(c)} \lambda_{ij} \cdot \text{delay}_{ij}$, incorporating HBT delay when nets span across dies. The cell is finally moved to the position with the minimum cost.

3.4 Timing-driven 2.5D Global Placement

After our 3D placement, all instances are partitioned across dies with detailed positions. We then perform HBT insertion by placing each HBT at the center of the wirelength-minimized region for its corresponding net according to Equation (1). To further enhance placement quality, we also adopt a 2.5D global placement stage to co-optimize positions of instances and HBTs by minimizing die-to-die wirelength, where a net-weighting strategy is used to guide the optimization process. To solve non-overlapping constraints of instances and pitch

Algorithm 1 Timing-driven 3D Detailed Placement

Input: 3D Global Placed netlist, timing graph $G = (V, E)$
Output: Timing-optimized placement solution $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{d}$

```
Initialize Lagrangian multipliers λ_ij;                          ▷ Solve LDP
for iter = 1 to K do
    Sort cells by λ value of output pin in descending order;
    Find maximal independent set S;
    cost_best ← +∞;
    for each cell c ∈ S in parallel do                           ▷ Solve LRS/λ
        for each candidate position p in search window do
            Move c to p and update net delays incrementally;
            cost ← Σ_{(i,j)∈E_net(c)} λ_ij · delay_ij;
            if cost < cost_best then
                pos_best ← p, cost_best ← cost;
        Move c to pos_best;
    Update Lagrangian multipliers λ_ij;                          ▷ Solve LDP
```

Table 1: Statistics of the Open3DBench benchmarks. All the netlists are optimized by the 2D *repair_design* command to repair violations. r_{macro} denotes the macro area ratio.

Design	#Macros	#Cells	#Nets	#IOs	r_{macro}	Period(ns)
ariane133	132	169081	184873	495	0.53	3
ariane136	136	173930	189136	495	0.53	3
bp	24	299465	331544	1198	0.40	2
bp_be	10	51132	58585	3029	0.56	2.6
bp_fe	11	32934	35985	2511	0.71	1.8
bp_multi	26	145916	160403	1453	0.57	0.8
bp_quad	220	1235609	1428521	135	0.48	5.2
swerv_wrapper	28	103433	112160	1416	0.67	2

Table 2: The parameters of hybrid bonds based on [39, 40].

Hybrid Bond Parameters	Pitch	Height	Width	Resistance	Capacitance
	2μm	1μm	1μm	3Ω	0.6fF

constraints of HBTs, a die-by-die 2D density penalty is applied for the top die, bottom die and HBT layer, respectively. The overall objective can be formulated as follows:

$$\min_{\boldsymbol{p}^+, \boldsymbol{p}^-, \boldsymbol{p}'} \sum_{e \in E} \bar{w}_e \cdot \text{WL}_e + \langle \boldsymbol{\lambda}, \boldsymbol{D} \rangle, \tag{18}$$

where $\boldsymbol{p}^+$, $\boldsymbol{p}^-$, and $\boldsymbol{p}'$ are the positions of the instances on the top die, bottom die, and HBT layer, respectively. $\text{WL}_e = \text{HPWL}_{e^+} + \text{HPWL}_{e^-}$ is the die-to-die wirelength of net e. $\bar{w}_e$ is computed using the same net weight calculation method as in 3D global placement with momentum-based update. $\boldsymbol{D} = (\boldsymbol{D}^+, \boldsymbol{D}^-, \boldsymbol{D}')$ is the density vector and $\boldsymbol{\lambda}$ is the density penalty vector.

3.5 2.5D Legalization and Detailed Placement

In this stage, we perform legalization and detailed placement for the instances and HBTs die-by-die. For legalization, we adopt transitive closure graph (TCG)-based macro legalization [36], and employ row-based algorithms including Abacus [37] and Tetris [38] for standard cells legalization. HBTs are legalized with the same padding strategy in [12] due to the pitch constraint. After legalization, we run traditional 2D detailed placement with ABCDPlace [35].

4 Experimental Results

All experiments are conducted on the Open3DBench benchmarks [18], which provides a complete 3D cell library and supports a full open-source flow from placement to routing with 3D STA. The detailed benchmark statistics are shown in Table 1. We do not consider the contest cases due to the lack of corresponding 3D cell library. The physical parameters of hybrid bonds are listed in Table 2. We implement

Table 3: The TNS ($\times 10^5$) ps and WNS ($\times 10^3$) ps results of comparison with baselines. 3D DP denotes the 3D detailed placement.

Design	Open3DBench [18]			Contest 1st Place [14]			[12]			[11]			Ours wo. 3D DP			Ours		
	TNS	WNS	Time (s)	TNS	WNS	Time (s)	TNS	WNS	Time (s)	TNS	WNS	Time (s)	TNS	WNS	Time (s)	TNS	WNS	Time (s)
ariane133	-436.45	-7.13	54.00	-83.38	-3.32	195.79	-90.28	-3.46	222.42	-97.25	-3.24	59.45	-17.20	-1.03	320.22	**-0.43**	**-0.71**	781.30
ariane136	-516.92	-9.92	55.00	-66.58	-2.67	229.47	-67.56	-2.71	255.23	-145.43	-3.81	63.68	-13.02	-1.09	100.24	**-10.94**	**-0.75**	583.24
bp	-626.78	-11.85	93.00	-39.33	-5.17	413.75	-50.92	-7.50	455.86	-89.50	-7.04	88.84	-77.99	**-4.62**	174.44	**-2.23**	-4.70	880.32
bp_be	-160.69	-23.90	51.00	-0.27	-0.94	108.84	-0.26	-0.70	113.69	-3.28	-1.86	30.32	-0.15	**-0.23**	41.68	**-0.02**	-0.39	242.76
bp_fe	-140.76	-7.24	47.00	-14.16	-1.55	106.82	-14.05	-1.41	105.78	-104.55	-8.10	25.17	-4.52	-1.09	34.55	**-3.20**	**-0.91**	191.51
bp_multi	-296.02	-8.69	59.00	-87.31	-6.95	266.56	-87.76	-6.95	282.36	-85.89	-6.68	52.13	-79.06	**-5.34**	218.90	**-36.49**	**-5.34**	959.80
bp_quad	-5873.29	-23.04	399.00	-1651.75	-11.48	1538.95	-1778.96	-11.96	1736.54	-623.61	-3.89	166.15	-356.44	**-1.62**	641.74	**-312.66**	-1.66	2830.21
swerv_wrapper	-528.18	-6.73	76.00	-154.06	-3.13	182.24	-162.24	-3.18	195.71	-369.15	-4.65	43.18	**-121.92**	-2.22	64.72	-133.87	**-2.12**	361.86
Avg. Ratio	1001.58	14.21	0.14	30.47	2.89	0.43	33.15	2.93	0.46	56.95	3.82	0.10	11.05	1.09	0.22	**1.00**	**1.00**	1.00

(a) (b) (c)

Figure 7: 3D placement of bp_quad: (a) top die, (b) bottom die, and (c) HBT layout of our timing-driven placer.

the whole framework in C++ and CUDA based on the open-source placer DREAMPlace [31] with GPU acceleration. Gurobi [41] is used as the MILP solver. For all the cases, the HBT cost weight parameter α is set to 1.5 to achieve a reasonable HBT density. Through grid search, the wirelength weight parameter β is set to 0.01, the distance criticality parameter σ is set to 2.0 and τ is set to 3.0. By default, the z-axis bin number is set to 32 and the bin size is set to the average of x-axis and y-axis bin sizes. All experiments are performed on a Linux server equipped with 24 Intel Xeon Silver 4310 cores, one NVIDIA GeForce RTX 4090 GPU, and 64 GB of RAM. We compare our framework against several baselines, including the Open3DBench native placement flow *Open3D-DMP*, the 1st place solution from the ICCAD 2023 contest [14], and two state-of-the-art academic works [11, 12]. We evaluate the timing metrics using the Open3DBench STA engine and PDK after global routing stage. Since the Open3DBench PDK is built based on Nangate45 [42], a 45nm technology node, we scale the per-unit-length resistance of each metal layer by 10× to approximate advanced process nodes. This scaling increases the proportion of interconnect delay in critical paths, better reflecting the timing characteristics of modern sub-10nm technologies. By default, we turn on all the weighting strategies only after 500 iterations in global placement since the timing information in the early iterations is not reliable. We set 3D detailed placement iterations K to 10 and choose the solution with the best timing performance. The size of the discrete search space is set to $(5 \times \text{site\_width}) \times 200 \times 2$.

Table 3 presents the comparative results of our framework against four baselines. These results were obtained by executing their binary files on our machine. The results reveal that our framework achieves the best results on all the cases, achieving 30.47 ∼ 1001.58× reduction in TNS and 2.89 ∼ 14.21× reduction in WNS. Since the Open3DBench native flow only supports memory-on-logic placement, it is not surprising that it performs poorly compared to the other logic-on-logic approaches due to a limited solution space. Compared to the ICCAD 2023 contest winner and two academic works, our placer, both with and without 3D detailed placement configurations, consistently outperforms them, demonstrating the effectiveness of our proposed timing-driven optimizations. Since the timing-driven placement needs to iteratively perform STA and other operations such as weighting calculation, our placer sacrifices a certain amount of runtime compared to the baselines. The runtime bottleneck of our framework is the 3D detailed placement due to the large discrete 3D search space. However, the overall runtime is acceptable and is justified by the significant timing improvements. These improvements demonstrate that the 3D design space still offers rich opportunities for timing optimization.

Table 4: The WNS ($\times 10^3$) ps and TNS ($\times 10^5$) ps results of the ablation study. The best results are highlighted in bold.

Design	Ours wo. weighting		Ours wo. 3D DP		Ours	
	TNS	WNS	TNS	WNS	TNS	WNS
ariane133	-28.99	-1.42	-17.20	-1.03	**-0.43**	**-0.71**
ariane136	-64.25	-2.6	-13.02	-1.09	**-10.94**	**-0.75**
bp	-13.67	-5.35	-77.99	**-4.62**	**-2.23**	-4.7
bp_be	-0.03	**-0.23**	-0.15	**-0.23**	**-0.02**	-0.39
bp_fe	-15.54	-1.6	-4.52	-1.09	**-3.20**	**-0.91**
bp_multi	-39.52	-5.88	-79.06	**-5.34**	**-36.49**	**-5.34**
bp_quad	-449.03	-4.38	-356.44	**-1.62**	**-312.66**	-1.66
swerv_wrapper	-219.72	-2.77	**-121.92**	-2.22	-133.87	**-2.12**
Avg. Ratio	11.27	1.75	11.05	1.09	**1.00**	**1.00**

Table 5: #HBTs used in comparison with [12].

Design	#HBTs	
	[12]	Ours
ariane133	42284	65122
ariane136	51265	68933
bp	44564	79717
bp_be	14280	20202
bp_fe	5275	11863
bp_multi	25514	49244
bp_quad	265289	420407
swerv_wrapper	27139	41661
Avg. Ratio	0.61	1.00

We further conduct ablation studies to assess the impact of our net and pin weighting strategies in 3D and 2.5D global placement, and our 3D detailed placement, as summarized in Table 4. The data shows 11.27× (1.75×) and 11.05× (1.09×) improvement on TNS (WNS), respectively, compared to configurations without weighting strategies and 3D detailed placement. These results demonstrate that both components substantially improve final timing quality. The placement layout of bp_quad is shown in Figure 7, including both top die, bottom die and HBT layout. It is shown that our placer achieves a balanced area distribution between the top and bottom dies with high projection overlap. Table 5 shows that our placer averages 39% more HBTs than [12] over all the cases. Nevertheless, this HBT density is manufacturable by advanced processes such as TSMC SoIC™ [3].

5 Conclusion

This paper presents the first timing-driven true-3D placement framework for hybrid-bonding-based F2F 3D ICs. We develop a native 3D STA engine that models hybrid bond parasitics in distributed RC networks, and integrate a sample-based 3D wirelength model with hybrid net-pin weighting for timing-driven 3D global placement, along with Lagrangian-based 3D detailed placement for fine-grained timing optimization. Experimental results on Open3DBench demonstrate substantial improvements over state-of-the-art 3D placers, achieving 30 ∼ 1002× reduction in TNS and 3 ∼ 14× reduction in WNS, confirming that native timing-driven placement unlocks significant optimization opportunities for 3D designs.

References

[1] ITRS, "International technology roadmap for semiconductors 2.0," Semiconductor Industry Association, Technical Report, 2015, accessed: 2025-10-19. [Online]. Available: https://www.semiconductors.org/resources/2015-international-technology-roadmap-for-semiconductors-itrs/

[2] T. F. Wu, H. Liu, H. E. Sumbul, L. Yang, D. Baheti, J. Coriell, W. Koven, A. Krishnan, M. Mittal, M. T. Moreira, M. Waugaman, L. Ye, and E. Beigné, "11.2 a 3D integrated prototype system-on-chip for augmented reality applications using face-to-face wafer bonded 7nm logic at <2μm pitch with up to 40% energy reduction at iso-area footprint," in *Proc. ISSCC*, vol. 67, 2024, pp. 210–212.

[3] D. C. H. Yu, C.-T. Wang, and H. Hsia, "Foundry perspectives on 2.5D/3D integration and roadmap," in *Proc. IEDM*, 2021, pp. 3.7.1–3.7.4.

[4] J. Wuu, R. Agarwal, M. Ciraula, C. Dietz, B. Johnson, D. Johnson, R. Schreiber, R. Swaminathan, W. Walker, and S. Naffziger, "3D V-Cache: the implementation of a hybrid-bonded 64MB stacked cache for a 7nm x86-64 CPU," in *Proc. ISSCC*, vol. 65, 2022, pp. 428–429.

[5] C. Li, Y. Yin, X. Wu, J. Zhu, Z. Gao, D. Niu, Q. Wu, X. Si, Y. Xie, C. Zhang, and G. Sun, "H2-LLM: Hardware-dataflow co-exploration for heterogeneous hybrid-bonding-based low-batch LLM inference," in *Proc. ISCA*, 2025, pp. 194–210.

[6] Z. Fu, X. Guo, W. Zeng, S. Zhong, Y. Zhang, P. Chen, R. Wang, L. Ye, and M. Li, "H2EAL: Hybrid-bonding architecture with hybrid sparse attention for efficient long-context LLM inference," in *Proc. ICCAD*, 2025, pp. 1–7.

[7] Z. Yue, Y. Wang, C. Li, S. Wei, Y. Hu, and S. Yin, "3D-PATH: A hierarchy lut processing-in-memory accelerator with thermal-aware hybrid bonding integration," in *Proc. MICRO*, 2025, pp. 78–93.

[8] 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 TCAD*, vol. 36, no. 10, pp. 1716–1724, 2017.

[9] 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 TCAD*, vol. 39, no. 6, pp. 1151–1164, 2020.

[10] 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 TCAD*, vol. 43, no. 6, p. 1624–1637, 2024.

[11] Y. Zhao, P. Liao, S. Liu, J. Jiang, Y. Lin, and B. Yu, "Analytical heterogeneous die-to-die 3-D placement with macros," *IEEE TCAD*, vol. 44, no. 2, p. 402–415, 2025.

[12] 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 *Proc. DAC*, 2024, pp. 1–6.

[13] K.-S. Hu, I.-J. Lin, Y.-H. Huang, H.-Y. Chi, Y.-H. Wu, and C.-F. C. Shen, "2022 ICCAD CAD contest problem b: 3D placement with D2D vertical connections," in *Proc. ICCAD*, 2022, pp. 1–5.

[14] K.-S. Hu, H.-Y. Chi, I.-J. Lin, Y.-H. Wu, W.-H. Chen, and Y.-T. Hsieh, "Invited paper: 2023 ICCAD CAD contest problem b: 3D placement with macros," in *Proc. ICCAD*, 2023, pp. 1–6.

[15] Y. Shi, S. Xu, S. Kai, X. Lin, K. Xue, M. Yuan, and C. Qian, "Timing-driven global placement by efficient critical path extraction," in *Proc. DATE*, 2025, pp. 1–7.

[16] P. Liao, D. Guo, Z. Guo, S. Liu, Y. Lin, and B. Yu, "DREAMPlace 4.0: Timing-driven placement with momentum-based net weighting and lagrangian-based refinement," *IEEE TCAD*, vol. 42, no. 10, pp. 3374–3387, 2023.

[17] D. Kim, M. Kim, J. Hur, J. Lee, J. Cho, and S. Kang, "TA3D: Timing-aware 3D IC partitioning and placement by optimizing the critical path," in *Proc. MLCAD*, 2024, pp. 1–7.

[18] 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," 2025. [Online]. Available: https://arxiv.org/abs/2503.12946

[19] Z. Guo, T.-W. Huang, and Y. Lin, "GPU-accelerated static timing analysis," in *Proc. ICCAD*, 2020, pp. 1–9.

[20] ——, "Accelerating static timing analysis using CPU-GPU heterogeneous parallelism," *IEEE TCAD*, vol. 42, no. 12, pp. 4973–4984, 2023.

[21] T.-W. Huang and M. D. F. Wong, "UI-Timer 1.0: An ultrafast path-based timing analysis algorithm for CPPR," *IEEE TCAD*, vol. 35, no. 11, pp. 1862–1875, 2016.

[22] 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 *Proc. ICCAD*, 2020, pp. 1–9.

[23] P. Vanna-iampikul, J. Yoon, C. Park, G. Yeap, and S. K. Lim, "Placement-aware 3D net-to-pad assignment for array-style hybrid bonding 3D ICs," in *Proc. ISPD*, 2025, pp. 200–208.

[24] B. Fu, L. Liu, M. D. Wong, and E. F. Young, "Hybrid modeling and weighting for timing-driven placement with efficient calibration," in *Proc. ICCAD*, 2024, pp. 1–7.

[25] Y. Zhao, F. Gu, S. Liu, P. Liao, and B. Yu, "H3D: Heterogeneous resources aware global router for face-to-face bonded 3D ICs," in *Proc. ICCAD*, 2025, pp. 1–7.

[26] C. C. N. Chu and Y. Wong, "FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design," *IEEE TCAD*, vol. 27, no. 1, pp. 70–83, 2008.

[27] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *Journal of Applied Physics (JAP)*, vol. 19, no. 1, pp. 55–63, 1948.

[28] Z. Guo, H. Liu, X. Shi, S. Hua, Z. Zhang, C. Zhao, R. Wang, and Y. Lin, "HeteroSTA: A CPU-GPU heterogeneous static timing analysis engine with holistic industrial design support," in *Proc. ASPDAC*, 2026, pp. 1–7.

[29] J. Lu, H. Zhuang, I. Kang, P. Chen, and C.-K. Cheng, "ePlace-3D: Electrostatics based placement for 3D-ICs," in *Proc. ISPD*, 2016, pp. 11–18.

[30] J. Lu, H. Zhuang, P. Chen, H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng, and C.-K. Cheng, "ePlace-MS: Electrostatics-based placement for mixed-size circuits," *IEEE TCAD*, vol. 34, no. 5, pp. 685–698, 2015.

[31] Y. Lin, Z. Jiang, J. Gu, W. Li, S. Dhar, H. Ren, B. Khailany, and D. Z. Pan, "DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement," *IEEE TCAD*, vol. 40, no. 4, pp. 748–761, 2021.

[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 TCAD*, vol. 32, no. 4, pp. 497–509, 2013.

[33] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," 2017. [Online]. Available: https://arxiv.org/abs/1611.01144

[34] 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.

[35] Y. Lin, W. Li, J. Gu, H. Ren, B. Khailany, and D. Z. Pan, "ABCDPlace: Accelerated batch-based concurrent detailed placement on multithreaded CPUs and GPUs," *IEEE TCAD*, vol. 39, no. 12, pp. 5083–5096, 2020.

[36] 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 *Proc. ICCAD*, 2008, pp. 218–223.

[37] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Abacus: fast legalization of standard cell circuits with minimal movement," in *Proc. ISPD*, 2008, pp. 47–53.

[38] D. Hill, "Method and system for high speed detailed placement of cells within an integrated circuit design," Apr. 9 2002, US Patent 6,370,673.

[39] B. Ayoub, S. Lhostis, S. Moreau, E. L. Perez, J. Jourdon, P. Lamontagne, E. Deloffre, S. Mermoz, C. de Buttet, V. Balan, C. Euvard, Y. Exbrayat, and H. Frémont, "Impact of process variations on the capacitance and electrical resistance down to 1.44 μm hybrid bonding interconnects," in *Proc. EPTC*, 2020, pp. 453–458.

[40] J. H. Lau, "Recent advances and trends in Cu-Cu hybrid bonding," *IEEE TCPMT*, vol. 13, no. 3, pp. 399–425, 2023.

[41] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2023, accessed: 2023. [Online]. Available: https://www.gurobi.com/

[42] "NanGate 45nm." [Online]. Available: https://si2.org/open-cell-library/