Closing the Gap: Advantages of Block-Level over Gate-Level in 3D IC Design for Advanced Nodes

Min Gyu Park[1], Pruek Vanna-Iampikul[2] Sung Kyu Lim[1]
[1] University of Southern California, Los Angeles, USA; [2] Burapha University, Chonburi, Thailand;
email: parkming@usc.edu

***Abstract*—Gate-level 3D ICs have demonstrated substantial power and performance improvements over 2D ICs. However, at advanced technology nodes, achieving sufficient hybrid bond density within a reduced chip footprint requires a 200nm bond pitch—beyond the limits of current manufacturing capabilities. To address this issue, we focus on block-level 3D IC design which has fewer top-level connections. Previous methodologies for block-level 3D IC design suffer from several limitations: (1) reliance on slow simulated annealing-based floorplanning, (2) suboptimal 3D design flows lacking post-route optimization, (3) a lack of attention to the critical initial step of soft block design, and (4) excessive hybrid bond usage that necessitates a 500nm pitch. To overcome these challenges, we propose a comprehensive methodology that includes: (1) a fast, gradient-based analytical solver, (2) Fence-3D flow, delivering up to 15% improvement in power-delay product, (3) ML-based congestion-aware soft block sizing, delivering up to 6.6% improvement, and (4) a partial-MLS method that selectively applies Metal Layer Sharing, reducing hybrid bond count by up to 71%. Collectively, these techniques enable the use of a 1um hybrid bond pitch in 3nm block-level 3D ICs.**

I. Introduction

Gate-level 3D ICs have emerged as a key approach of advanced packaging, offering significant advantages over their 2D counterparts. By transforming long interconnects in 2D into shortened vertical connections in 3D, they substantially reduce load capacitance, lowering power consumption while improving performance through shorter critical paths. To realize commercial-grade Face-to-Face (F2F) bonded Logic-on-Logic 3D ICs, extensive research has been conducted. Among the most representative methods is the Pin-3D [16] flows, which follow a pseudo-3D placement: it first performs physical design in 2D and subsequently partitions the design into two tiers.

Although Pin-3D, a dedicated design flow for gate-level 3D ICs, has demonstrated promising results at old technology nodes (e.g., 28nm and 16nm), it encounters significant challenges when scaled to advanced nodes. Specifically, due to reduced chip footprints, gate-level designs require an aggressive hybrid bond pitch of 200nm to accommodate sufficient vertical connections, as shown in Table I. However, current fabrication capabilities limit hybrid bond pitches to 1um, rendering such gate-level 3D designs infeasible.

To address this manufacturability gap, we shift our focus to block-level 3D ICs. While the pseudo-3D placement stage forces cells within a single module to be distributed across two tiers—resulting in a large number of tier-crossing connections—the block-level design assigns each module to a single tier. As a result, only inter-block connections can generate hybrid bonds, significantly reducing their total count. However, block-level designs suffer from degraded power and performance due to the separation of intra- and inter-block optimization. In this work, we identify not only the limitations of existing block-level 3D IC design methodologies but also address critical challenges that have been previously overlooked, presenting a suite of techniques that narrow the design quality gap with gate-level 3D ICs while enabling a practical 1um hybrid bond pitch at the 3nm node. Our key contributions are as follows:

TABLE I: Hybrid bond count in gate-level 3D ICs for the JPEG design. We highlight the maximum number across technology nodes and pitch sizes. Both 1um and 500nm pitches are infeasible for the 3nm node.

		28nm	16nm	3nm
3D chip size (μm)		528×528	317×317	102×102
# hybrid bonds used		~65,000		
Max.	pitch = $1\mu m$	278,784	100,489	10,414
h-bond	pitch = $500nm$	1,115,136	401,956	41,616
offered	pitch = $200nm$	6,969,200	2,512,225	260,100

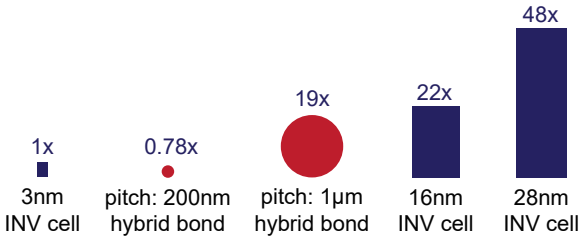

Fig. 1: Relative size comparison between hybrid bonds and INV cells.

- We propose an analytical 3D floorplanner that significantly outperforms previous simulated annealing-based approaches in runtime and scalability.
- To enable post-route optimization —previously unsupported in Pin-in-the-Middle [9] —we develop the Fence-3D flow, which achieves up to a 15% improvement in Power-Delay Product.
- For the first time, we investigate soft block sizing—an essential first step in block-level design that has remained largely unexplored—and propose an ML-based congestion-aware sizing technique, achieving a 6.6% improvement in PDP.
- Finally, we propose a partial-MLS method that selectively applies Metal Layer Sharing (MLS) to critical regions, reducing hybrid bond count by up to 71% compared to the Full-MLS.

II. Why Block-Level 3D IC in Advanced Nodes?

While technology nodes have significantly scaled down over the past decade, the minimum manufacturable hybrid bond pitch has remained relatively stagnant—limited to the 1um to 700nm [4]. This mismatch creates a substantial scale disparity between 3nm inverter cells and the 1um hybrid bond pitch, as illustrated in Fig.1, which in turn severely limits the feasible hybrid bond count, as quantified in Table I. As a result, gate-level 3D ICs become impractical at advanced nodes due to aggressive 200nm pitch requirements, whereas block-level 3D ICs—characterized by a smaller number of inter-block connections—present a viable path forward.

Moreover, advanced nodes increasingly suffer from reduced yield. This issue is further exacerbated in 3D ICs, where die-to-die inte-

Algorithm 1 Hierarchical Clustering for Soft Block Generation

Input: $\mathbf{N_{hier}}$: Synthesized netlist with hierarchical cell naming.
$\mathbf{D_{cellarea}}$: Dictionary mapping std cells to their respective areas.
$\mathbf{B_{hard}}$: List of hard blocks, including memory macros and IPs.
$\mathbf{A_{range}}$: List of min. and max. allowable areas for soft blocks.
Output: $\mathbf{G_{blocks}}(\mathbf{V}, \mathbf{E})$: Block-level graph.

```
function ClusterModule(module)
    area ← Compute module area via D_cellarea
    if area > A_range[max] then
        for sub in {submodules of module} do
            ClusterModule(sub)
    else if area ≥ A_range[min] then
        Add module as soft block to V

G_blocks(V, E) ← empty; add B_hard as hard block to V
ClusterModule(top module of N_hier)
Add block edges derived from cell links in N_hier to E
```

gration introduces additional sources of yield degradation compared to 2D designs. Block-level 3D ICs offer a promising solution to this challenge. By enabling module-level testability, defective dies can be identified and discarded prior to integration, significantly mitigating overall yield loss. In addition, the reduced number of hybrid bonds lowers the probability of failure during the bonding process.

III. Related Work and Shortcoming

Implementing block-level 3D ICs requires a dedicated floorplan methodology and design flow. Existing approaches to 3D floorplanning, such as [1], [5], [19], leverage sequence pair representations with simulated annealing. However, as shown in Table II, simulated annealing exhibits slow convergence due to its stochastic move selection mechanism. Furthermore, simulated annealing is less scalable [2], posing a significant limitation for industrial-scale designs that often involve 50 to 100 blocks. To address these challenges, we propose an analytical solver for 3D floorplanning that optimizes a differentiable loss function.

The current state-of-the-art block-level 3D IC design flow is Pin-in-the-Middle [9], which implements commercial-grade 3D IC layouts. It places block pins in the middle, achieving a 12% improvement in Energy-Delay Product (EDP) compared to 3D block-level designs where all pins are located on the block boundary. While this approach enhances the quality of block-level 3D IC design, it is still constrained by its reliance on the Compact-2D flow [11], which limits post-route optimization compared to Pin-3D flow [16].

IV. Our Block-level 3D Design Methodology

A. Hierarchical Clustering

Block-level design construction begins with clustering cells into soft blocks. To leverage the existing RTL hierarchy, we preserve hierarchical information during synthesis by executing the commands $change\_names$ *-rule verilog -hierarchy* followed by $ungroup$ *-all -*$flatten$ in sequence. Given a synthesized netlist with hierarchical naming and cell area information derived from the target tech node, Algorithm 1 performs hierarchical clustering of the netlist and generates an undirected graph consisting of multiple blocks and their connections.

Users can define both minimum and maximum area bounds for soft block generation. A higher upper bound enables the formation of larger soft blocks, thereby reducing the total number of blocks and, consequently, minimizing the number of required hybrid bonds. On the other hand, the lower bound is essential to filter out excessively small clusters—typically containing only 10–100 standard cells—that are better handled as gate-level instances during top-level integration. In this work, the max and min area bounds are set to (1,100 um^2, 100 um^2) for AES, (2,500 um^2, 100 um^2) for JPEG, and (4,100 um^2, 100 um^2) for Rocket DualCore, respectively. For each benchmark, the maximum area bound was determined by defining units that can be appropriately grouped at the functionality level.

TABLE II: Comparison between 3D floorplanning algorithms: Sequence Pair + Simulated Annnealing [1], [5], [19] vs. our Analytical Solver. Floorplanning runtime is measured when the output chip size difference between the two algorithms is within 5%.

	Sequence Pair + Annealing	Our Analytical
11 blocks	2h 12m	18min
16 blocks	3h 18m	25min
41 blocks	16h 10m	60min

Algorithm 2 Multi-Stage Analytical 3D Floorplanner

Input: $\mathbf{G_{blocks}}(\mathbf{V}, \mathbf{E})$: Block-level graph from Algorithm 1
RX, RY: Chip dimensions, lr: learning rate, max_{ap}: Maximum aspect ratio of soft block, $Adam_c$: Custom Adam optimizer ensuring block locations and widths remain within boundaries.
a_i: Weighting coefficient for each loss term.
Output: Optimized 3D floorplan.

```
λ ← λ_init                          ▷ λ_init to balance gradients of L1 and L2
opt ← Adam_c(RX, RY, lr, max_ap, β1, β2)
Place soft blocks centrally, with soft blocks initialized as squares.
Hard blocks are pre-placed using a commercial macro placer.
for s in stage do
    for i in epoch do
        L1 ← a1·L_HPWL + a2·L_Hbond + a3·L_Balance + a4·L_Z-penalty
        L2 ← λ · Density Control
        x_i, y_i, z_i, w_i ← opt(∇L1 + ∇L2) + F_repulsive
    λ ← λ · 2                       ▷ Increase density control weight
```

Each node in the block-level graph has an area attribute (A_i). For soft blocks, the area attribute A_i is calculated using a uniform placement density of 60%, which serves as the baseline. The 60% density is chosen to reserve space for additional buffer insertion during top-level optimization. In contrast, hard blocks retain their predefined area values. Each undirected edge is weighted based on the number of inter-block connections, which is determined by the total number of connections between cells within the respective blocks.

B. 3D Floorplanning

This section focuses on 2-tier 3D ICs, but can be extended to N-tier by dividing the z_i into N regions and adjusting the loss terms.

Problem: Given m mixed blocks, determine the optimal location (x_i, y_i, z_i) for each block, and the width w_i for soft blocks. Once w_i is specified, the height h_i is automatically determined as A_i/w_i.

Fixed outline constraint: The chip area for a two-tier 3D IC is defined as half the sum of all block areas (A_i) divided by a placement density of 0.7. The chip width (RX) and height (RY) are the square root of this area. Given the chip size, the constraints

on block placement are as follows:

$$x_i \in [0, RX - w_i], \quad y_i \in [0, RY - h_i], \quad z_i \in [0, 1] \quad \forall i \in [1, m]$$

Our loss function consists of five terms: differentiable HPWL, hybrid bond number, 3D tier area balancing, z-penalty, and density control for overlap reduction. For differentiable HPWL, we compute the wirelength using the center coordinates of each block, adopting the Weighted Average (WA) formulation [3]:

$$L_{\text{HPWL}} = \sum_{(i,j)\in\mathbf{E}} w_{ij} \cdot (\text{WA}(x_i + \tfrac{w_i}{2},\ x_j + \tfrac{w_j}{2}) + \text{WA}(y_i + \tfrac{h_i}{2},\ y_j + \tfrac{h_j}{2}))/(RX * RY)$$

To minimize the hybrid bond count, the bond loss is defined as:

$$L_{\text{Hbond}} = \sum_{i\neq j} \frac{1}{1 + e^{\alpha(z_i - 0.5)(z_j - 0.5)}} w_{ij}$$

where w_{ij} represents the edge weight between blocks i and j. This term serves as a smooth approximation of:

$$\sum_{i\neq j} \begin{cases} w_{ij}, & \text{if } (z_i - 0.5)(z_j - 0.5) < 0 \\ 0, & \text{otherwise} \end{cases}$$

It counts hybrid bonds when two distinct blocks are on opposite tiers relative to $z = 0.5$. The parameter α controls the steepness of the approximation, with $\alpha = 10$ used in this study.

Achieving area balancing between tiers is essential for a compact design. The area balancing term is formulated as:

$$L_{\text{Balance}} = \left(\sum_i z_i A_i - \sum_i (1 - z_i) A_i \right)^2 / (RX * RY)$$

However, this formulation tends to force all z_i values toward 0.5, requiring an additional z-penalty term that drives z_i away from 0.5:

$$L_{\text{Z-penalty}} = \sum_i z_i (1 - z_i)$$

Similar to analytical placement, overlap reduction techniques are required to mitigate block overlaps. We adopt a multi-stage density control strategy with bell-shaped smoothing, as summarized in [2], combined with a repulsive force that alleviates unresolved overlaps among blocks clustered at the corners due to fixed-outline constraints. This force is inversely proportional to the squared distance between overlapping blocks and is bounded to stabilize optimization:

$$F^{(i,j)}_{\text{repulsive,x}} = \text{clip}\left(\frac{1}{(x_i - x_j)^2 + \varepsilon}, -\log RX, \log RX \right) / RX$$

The complete analytical 3D floorplan methodology is outlined in Algorithm 2. Another key advantage of the proposed solver is its minimal need for hyperparameter tuning. Since the loss terms are normalized to be independent of chip size, the coefficients a_i can be set at a glance to keep the initial magnitudes of all loss components comparable. In contrast, sequence pair-based floorplan with simulated annealing requires careful hyperparameter tuning.

C. Pin and Fence Management

To achieve full post-route optimization capabilities, we propose the Fence-3D flow, as illustrated in Fig. 2. There are two key differences between the Fence-3D and Pin-3D flows. Since Fence-3D targets block-level design, it does not require tier-partitioned pseudo-3D placement as input. Instead, 2D blocks are generated on dedicated top and bottom tiers based on the hierarchical clustering result and 3D floorplan. Once all blocks are implemented using a standard 2D physical design flow, they are assembled using the $assembleDesign$ command in Cadence Innovus. This approach ensures seamless block-level optimization while preserving block boundaries and allowing mid-block pin assignment, following the same method as Pin-in-the-Middle [9].

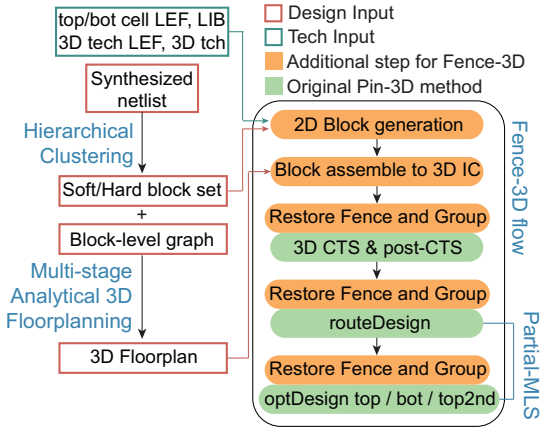

Fig. 2: Proposed design methodology for block-level 3D ICs. Partial-MLS can be supported by modifying the steps after the post-CTS stage.

TABLE III: Comparison of two different physical design methods for block-level 3D ICs: Pin-in-the-Middle (PITM) [9] and Fence-3D.

Benchmark	AES		JPEG	
3D design flow	PITM [9]	Fence-3D	PITM [9]	Fence-3D
Power (mW)	75.5	74.1	301.9	292.1
Perf. (GHz)	4.39	4.93	3.31	3.78
PDP (fJ)	17	15 (-13%)	91	77 (-15%)

During the iterative physical design process, the status of cells and macros alternates between CORE (for cells), BLOCK (for macros), and COVER. Consequently, the def file loses critical information on block fences and group constraints for components on the opposite die, which are in the COVER status. To address this issue, the group and fence information is preserved and is restored when transitioning from the COVER back to the CORE and BLOCK. As a result, the Fence-3D flow retains the advantage of assigning pins within the block interior, as in Pin-in-the-Middle, while fully leveraging the optimization capabilities of Pin-3D for block-level 3D IC design. This is demonstrated in the Table III, which highlights the superior power and performance achieved by the Fence-3D flow.

D. ML-based Congestion-Aware Soft Block Sizing

1) Trade-offs in Soft Block Sizing: The first step in block-level IC design is the implementation of soft blocks. However, to date, no existing work has addressed this initial step in depth. While there are substantial works on automatic 3D floorplanning [5], [19] and 2D floorplanning [12], [17], these studies did not consider block sizing, as the commonly used MCNC benchmarks provide block sizes as input. Additionally, prior work on block-level 3D IC design, Pin-in-the-Middle [9], did not discuss the design of individual blocks, focusing instead on block integration methods.

Prior to the physical design of each block, several inputs must be defined, and in this study, we focus specifically on the placement density, which directly influences block size. Our baseline approach

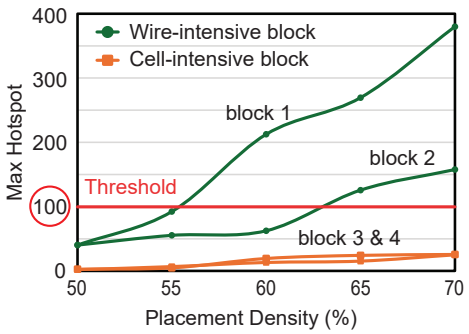

Fig. 3: Four representative blocks from the AES benchmark. Two of them are identified as wire-intensive, exhibiting maximum hotspot values exceeding the threshold of 100.

TABLE IV: Summary of the dataset used in this work. Build time for the training and validation sets includes both netlist generation (NG) and physical design (PD). Our dataset enables the model to generalize to unseen blocks without requiring additional data generation.

	Train & Valid set	Test set
Block instance count	[1,000, 10,000]	[400, 70,000]
# Cell-intensive blocks	743	36
# Wire-intensive blocks	257	20
# Total blocks	1000 (8:2 stratified split)	56
Data build time	9.7h (NG) + 24.3h (PD)	80.5h (PD)

adopts a uniform placement density of 60% for all soft blocks, as there is no available information on individual block characteristics at the RTL level. However, as shown in Fig. 3, we observed that certain blocks exhibit severe routing congestion, with a maximum hotspot value [1] approaching 400, even under high-effort congestion-driven placement. To mitigate this, a lower placement density is required to distribute cells and allocate additional routing resources. Conversely, we identified that some blocks do not encounter routing congestion even at 75% placement density. Based on these observations, we classify blocks into two categories:

- **Wire-intensive block**: A block that experiences routing congestion (maximum hotspot $>$ 100) at high placement densities, prior to DRC errors.
- **Cell-intensive block**: A block that maintains a maximum hotspot $<$ 100 until DRC errors are encountered.

This classification provides insight into the trade-offs associated with soft block sizing. Increasing placement density reduces block size, which in turn minimizes inter-block wire length, lowering load capacitance for improved power and shortening connections for enhanced performance. However, applying high placement density to wire-intensive blocks leads to severe routing congestion, resulting in detoured routing paths that degrade both power and timing.

2) Proposed ML-based approach: To optimize block-level design while addressing the trade-offs of soft block sizing, we leverage Machine Learning to classify blocks as wire-intensive or cell-intensive based on RTL-level inputs. Our model enables wire-intensive blocks to adopt a lower placement density, mitigating intra-block congestion, while cell-intensive blocks utilize a higher placement density, benefiting from reduced inter-block connection lengths due to their compact

[1]The maximum hotspot represents the contiguous normalized area of the GCELLs experiencing routing overflow. As a general rule of thumb, a value under 100 is considered tolerable.

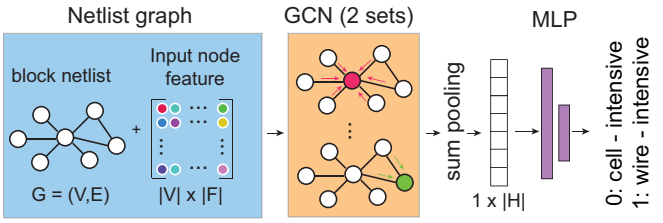

Fig. 4: Our ML model architecture consists of two Graph Convolutional Network (GCN) layers with sum pooling, followed by a Multi-Layer Perceptron (MLP) that reduces dimensions from 64 to 16 to 2.

TABLE V: Input node features in the netlist graph for each block.

Features	Descriptions
is DFF	1 for sequential cells, 0 for others
# fanout	Number of fanout of each cell
area	Area of the cell
drv strength	Driving strength of the cell (x1, x2, ..)
min. dist. to DFF	Shortest distance to DFF
max. dist. to DFF	Longest distance to first encountered DFF

size. The proposed ML model is expected to significantly reduce design time in industrial-scale block-level designs, while providing a valuable starting point for block optimization. This section details the dataset used, the ML model architecture, and the classification results.

As a netlist (= block) can be represented as a graph, a Graph Neural Network (GNN) is employed to handle this as a graph classification task. In this context, each soft block corresponds to a single data point, implying that constructing a large dataset necessitates a diverse set of soft blocks. To ensure the model generalizes well to unseen blocks, the training set must be sufficiently large and diverse while excluding blocks from the AES, JPEG, and Rocket DualCore benchmarks, which are designated as the test set. To achieve this, the Artificial Netlist Generator (ANG) [8] is utilized to generate training and validation datasets. ANG generates netlists by leveraging topological parameters and applying a distribution-matching algorithm for a given technology node. Prior study [8] has demonstrated that netlists generated by ANG exhibit similar characteristics to real circuits, achieving competitive classification performance and comparable distributions in t-SNE plots.

A total of 1,000 soft blocks are generated by randomly sampling topological parameters, including the number of instances (1,000–10,000), average net degree (2.5–4.0), timing path depth (5–15), and combinational logic ratio (20%–80%). Once the netlists are generated, a 2D physical design (PD) is performed for each block to determine the ground truth label, with placement density ranging from 50% to 80% in 5% increments and target frequencies varying from 1.0 GHz to 9.0 GHz in 0.5 GHz increments. Among the entire physical design results, blocks exhibiting routing congestion before DRC errors are classified as wire-intensive, while the remaining blocks are classified as cell-intensive. The results reveal that 257 blocks are wire-intensive, and the dataset is split into an 8:2 ratio for training and validation through stratified sampling. For the test set, 56 soft blocks extracted from the AES, JPEG, and Rocket DualCore benchmarks are analyzed using the same methodology, with 20 blocks identified as wire-intensive. The overall data construction time on three servers, each powered by a 2.10-GHz Intel® Xeon® Gold 6230 processor with 64 cores, is provided in Table IV.

The key distinction from previous ML-EDA works [13], [14] lies in

TABLE VI: Ablation study on (1) the effectiveness of distance features and (2) the choice of graph networks.

(1) Impact of Input Features (with GCN)		
	Validation Accuracy	Test Accuracy
w/o Distance Features	83.8%	83.1%
w/ Distance Features	95.3%	92.3%
(2) Performance Across Graph Networks		
GCN	95.3%	92.3%
GAT	92.3%	86.3%
Graph Transformer	96.2%	70.6%

utilizing pre-physical design inputs, rather than intermediate physical design results. This approach limits the use of detailed input features, such as timing, power, and capacitance. Consequently, extracting meaningful features at the netlist level becomes crucial. We configure the input node features as shown in Table V. Among these features, the shortest distance to DFF and the longest distance to the first encountered DFF play a critical role in improving accuracy. These features help the model learn how many long nets exist in each block, contributing to the wire-intensive characteristics. As demonstrated in Table VI, the inclusion of these features increases the classification accuracy from 83% to 92% for the test set.

The ML model is constructed as shown in Fig. 4. Binary classification is performed using Cross Entropy Loss, with a batch size of 32, 500 epochs, and a learning rate of 0.001, employing the Adam optimizer with β_1 = 0.9 and β_2 = 0.999. The model with the highest accuracy on the valid set during training is used to predict the test set. Various graph network architectures, including GCN [10], GAT [18], and Graph Transformer Network [20], are compared. As shown in Table VI, increasing network complexity degrades the model's ability to generalize from artificial netlists to real one, where it tends to predict all blocks as either wire-intensive or cell-intensive. The best performance is achieved with GCN, which is therefore selected as the graph network for this study.

E. Metal Layer Sharing in Block-level 3D ICs

1) Full-MLS and No-MLS method: Metal Layer Sharing (MLS) was first introduced in [15], enabling 2D nets—those connecting cells within the same tier—to utilize metal layers from the opposite tier in 3D ICs. A prior study has demonstrated the benefits of MLS in reducing manufacturing costs by minimizing the number of metal layers and improving performance by providing greater routing flexibility through an expanded set of metal layer options. Although it has primarily focused on Monolithic 3D ICs, these advantages remain applicable to F2F 3D ICs. Following the prior study, we define the approach that allows unrestricted use of MLS without additional constraints in the physical design flow as the Full-MLS method.

However, in the Full-MLS method, up to 77% of the total hybrid bonds are consumed by 2D MLS nets, limiting the use of a 1um pitch. This incurs not only integration costs but also additional verification overhead [6] associated with 2D MLS nets. To address this issue, we investigate the No-MLS method, which completely disables MLS. The implementation of the No-MLS method is illustrated in Fig. 5-(b). First, the $routeDesign$ step is divided into two phases: one for 2D net routing, where routing blockages are applied to the opposite tier, and another for 3D net routing. Second, routing blockages are introduced on the hybrid bond layers during post-route optimization. Finally, any 2D MLS nets generated in the previous steps are re-routed with routing blockages enforced on the opposite tier. Through this process, MLS nets are effectively eliminated.

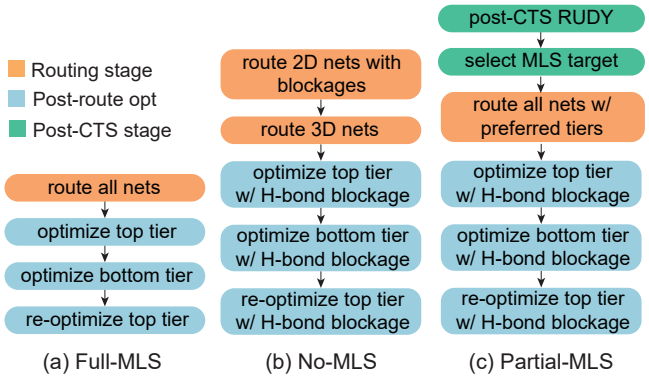

Fig. 5: Three variants of Metal Layer Sharing (MLS): (a) Full-MLS [15] allows unrestricted sharing across all nets, (b) No-MLS prohibits MLS entirely, and (c) Partial-MLS selectively enables MLS for critical nets to maintain performance while minimizing hybrid bond count.

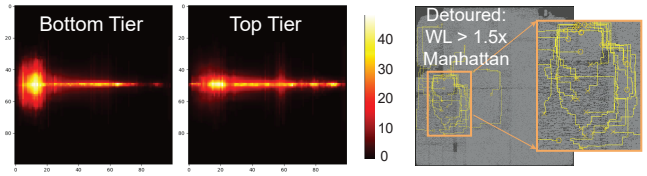

Fig. 6: Post-CTS RUDY map and detoured nets in the final GDS layout under the No-MLS scenario. The spatial correlation confirms the accuracy of RUDY-based congestion estimation.

2) Partial-MLS method: The Partial-MLS method aims to enable a 1um hybrid bond pitch in block-level 3D ICs by significantly reducing the number of hybrid bonds used for MLS while minimizing performance loss. First, RUDY maps are extracted under the No-MLS assumption, ensuring that congestion estimation reflects the worst-case scenario without MLS support. The congestion contribution of a net e within its bounding box is computed as follows:

$$RUDY_e(x,y) \propto \mu_e \left(\frac{w_e + h_e}{w_e h_e} \right) \propto \mu_e \left(\frac{1}{w_e} + \frac{1}{h_e} \right)$$

where w_e and h_e represent the width and height of the bounding box for net e, respectively. After computing the contribution of each net, RUDY maps are derived using the following formulation:

$$RUDY_{(m,n)} = \sum_{e \in \text{tile}_{mn}} \left(\frac{1}{w_e} + \frac{1}{h_e} \right) \frac{\text{Area}_{\text{tile} \cap \text{bbox}}}{\text{Area}_{\text{tile}}} \times C_e,$$

$$C_e = \begin{cases} 1, & \text{if } e \text{ is a 2D net} \\ 0.5, & \text{if } e \text{ is a 3D net} \end{cases}$$

C_e ensures that 2D nets contribute to the RUDY map of each tier, while 3D nets contribute to both maps with half the weight.

Once the RUDY map is extracted as shown in Fig.6-(a), the MLS candidate nets are selected from congested regions. Specifically, nets contributing to the top N% (set to 5% in this study) of congested regions are chosen for MLS support. These selected nets are then guided through the routing process using preferred layers, assigned from M5 bot to M5 top, to enable MLS during both global and detailed routing phases. The remaining 2D nets are assigned preferred layers from M5 bot to M6 bot for bottom 2D nets, and from M6 top

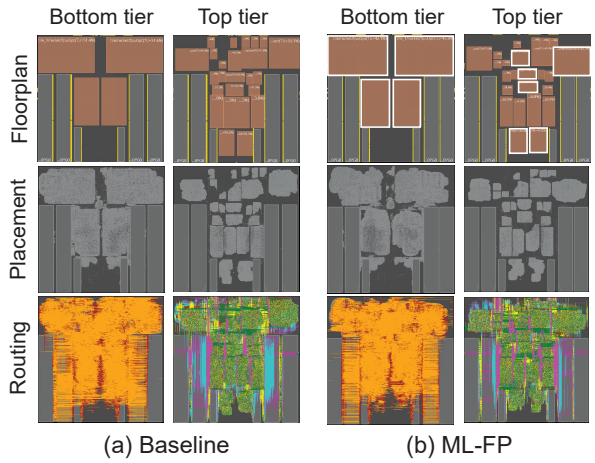

Fig. 7: GDS layouts of the Rocket DualCore benchmark with the Fence-3D flow: (a) The baseline design adopts a uniform 60% placement density across all blocks, while (b) the proposed ML-based floorplanner assigns a reduced 50% density to the wire-intensive block (highlighted in white) to mitigate routing congestion and a 70% to the cell-intensive block.

to M5 top for top 2D nets. The overall routing sequence follows the order: 3D nets → bottom/top 2D nets → 2D MLS candidates, ensuring that MLS is applied where congestion is most critical. Importantly, not all selected candidates are ultimately routed with MLS. By leveraging preferred layer constraints, the tool selectively applies MLS only to nets that traverse through congested regions, avoiding unnecessary MLS usage for nets merely originating or terminating within those areas. For example, in the JPEG benchmark, 212 out of 133,933 bottom nets and 329 out of 181,557 top nets are identified as MLS candidates. Among the total 541 MLS candidates, only 340 nets utilize MLS. After global and detail routing, it will follow the post-route optimization stage with routing blockages on the hybrid bond layers. The entire process is fully automated using TCL, $Bash$, and $Python$.

V. Experimental Results

A. Experimental Setup

With 3nm technology PDK, our design targets Homogeneous F2F bonded Logic-on-Logic block-level 3D ICs. The 3D BEOL consists of 12 metal layers, with 6 layers on the bottom tier (M1 bot, M2 bot, ..., M6 bot) and 6 layers on the top tier (M6 top, M5 top, ..., M1 top), where the layer names are ordered from the bottom to the top. Four metal layers are dedicated to intra-block routing for both the top and bottom tiers, while the remaining M5 bot, M6 bot, M6 top, and M5 top layers are allocated for inter-block connections. To ensure a fair comparison, we adopt a 200nm hybrid bond pitch, as the Full-MLS method is incompatible with 500nm and 1um pitch.

We evaluate three benchmarks which have hierarchical RTL structures: AES and JPEG, representing pure-logic, and Rocket DualCore, which incorporates memory. AES consists of 131k standard cells and is organized into 11 soft blocks through hierarchical clustering. JPEG is composed of 279k cells and is structured into 16 soft blocks. Rocket DualCore is built with 350k cells, incorporating 29 soft blocks and 12 hard memory macros. These benchmarks collectively span 11–41 blocks, effectively covering a wide design spectrum. Netlist synthesis is performed using Synopsys Design Compiler, while the Fence-3D physical design flow is implemented in Cadence Innovus, with power and timing analyses conducted in Cadence Tempus.

TABLE VII: Analysis of two representative blocks of designs in Fig. 7. With ML-FP, cell-intensive blocks achieve compact size with acceptable hotspot levels, while wire-intensive blocks alleviate routing congestion.

Blocks from Rocket DualCore		Baseline	ML-FP
Converter block	Type	cell-intensive	
	Placement density	60%	70%
	Block area (μm^2)	120	103
	Max. hotspot	5.6	24.3
FPU block	Type	wire-intensive	
	Placement density	60%	50%
	Block area (μm^2)	3,211	3,866
	Max. hotspot	214.5	12.5

TABLE VIII: Comparison of top-level quality metrics for designs in Fig. 7. Both utilize the Fence-3D with a 200nm hybrid bond pitch.

Rocket DualCore, 1.9GHz	Baseline	ML-FP
Chip size ($\mu m \times \mu m$)	160x160	163x164 (+4.4%)
Wire length (m)	1.055	1.084 (+2.7%)
# Hybrid bonds used	48,407	48,813
# of overflow G-cells	113	52 (-54%)
Power (mW)	194.2	194.4 (+0.1%)
Performance (GHz)	1.68	1.80 (+7.1%)
PDP (fJ)	115.6	108 (-6.6%)

B. State-of-the-Art Used for Comparison

This section summarizes the design methodologies employed throughout the study. Section V-C adopts the Fence-3D flow with a 200nm hybrid bond pitch to evaluate the impact of ML-based soft block sizing. Section V-D evaluates three different MLS control strategies, all under a consistent design flow that employs Fence-3D and ML-based soft block sizing. Section V-E and V-F compare gate-level and block-level 3D ICs using Pin-3D [16], Pin-in-the-Middle (PITM) [9], and Fence-3D. While Pin-3D and PITM adopt Full-MLS as in their original methodologies, our Fence-3D utilizes Partial-MLS. Each design employs the most relaxed hybrid bond pitch applicable to its flow—200nm for Pin-3D, 500nm for PITM, and 1um for Fence-3D. Additionally, since PITM does not incorporate soft block sizing techniques, it assumes a uniform 60% density, whereas Fence-3D leverages our ML-based soft block sizing.

C. Impact of ML-based Soft Block Sizing

Two block-level 3D ICs are compared: a Baseline design, where all soft blocks use a uniform placement density of 60%, and a ML-FP design, which leverages the ML model implemented in Section IV-D to classify soft blocks based on their characteristics. In the ML-FP approach, cell-intensive blocks are assigned a 70% density, while wire-intensive blocks are assigned a 50% density. To ensure that the floorplan does not affect design quality, both approaches share the same 3D floorplan. Fig. 7 illustrates the layouts of Rocket DualCore for both the Baseline and ML-FP designs, confirming that the block floorplan remains consistent between the two methods, with only slight adjustments to accommodate the modified soft block sizes.

As shown in the Table VII, the ML-based approach enables a compact implementation of cell-intensive blocks while maintaining a reasonable max hotspot. For wire-intensive blocks, it mitigates severe routing congestion by enlarging the block area. For Rocket DualCore, wire-intensive blocks like the FPU and Core consist of large soft

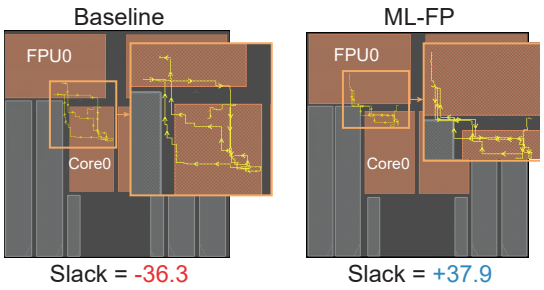

Fig. 8: Timing-critical path comparison for designs in Fig. 7. We show the critical path in the baseline vs. the same path in the ML-FP optimized.

TABLE IX: Critical path analysis of the timing paths in Fig. 8. A higher routed-to-Manhattan path length ratio indicates greater routing detour. ML-FP enables shortcut paths by mitigating intra-block congestion.

Rocket DualCore	Baseline	ML-FP
Max hotspot on Core block	647.2	45.1
Max hotspot on FPU block	214.5	12.5
Manhattan path length (μm)	256.6	236.6
Routed path length (μm)	340.8	278.8
Ratio (Routed to Manhattan)	x1.33	x1.18
Signal delay (ns)	555.7	485.1
Clock launch - capture (ns)	9.5	4.7

blocks, which increase chip size, wire length, and power consumption under ML-FP. However, routing congestion—measured by overflow GCells—is reduced by 54%, boosting performance and yielding a 6.6% PDP improvement as presented in the Table VIII. On the other hand, in the JPEG benchmark (result table is omitted due to space limits), most large soft blocks are cell-intensive. Therefore, ML-FP reduced chip size and wire length, achieving 1.8% power savings. Additionally, alleviated routing congestion improves performance by 4.1%, yielding a 5.8% overall PDP gain.

Through experiments on two benchmarks with distinct characteristics, we observe the following: (1) The impact of ML-FP on chip size and power is determined by the composition of soft blocks. (2) Effective intra-block congestion-aware soft block sizing serves as a key factor in full-chip performance improvement.

To investigate how intra-block congestion mitigation influences full-chip performance gain, a comparison is made between the negative slack timing path of the Rocket DualCore Baseline and the corresponding path extracted from ML-FP, as illustrated in Fig. 8. This critical path traverses the wire-intensive FPU and Core blocks, and as summarized in Table IX, intra-block congestion is significantly mitigated in ML-FP. When comparing the two paths, it is observed that the ratio of the actual routed path length to the theoretically ideal Manhattan path length is considerably lower in ML-FP than in the baseline. This indicates that, due to reduced congestion in ML-FP, the routing detour is avoided, leading to a shorter signal path and a reduction in signal delay, effectively lowering the negative slack.

D. Impact of Metal Layer Sharing

The impact of three MLS strategies is evaluated across all benchmarks, with the results summarized in Table XI. These implementations leverage the Fence-3D flow and ML-FP method. Since 500nm hybrid bond pitch requires a via legalization due to the size mismatch between hybrid bonds and vias [7], a 200nm pitch is used to eliminate its effect on power and performance, enabling a fair comparison based solely on MLS control.

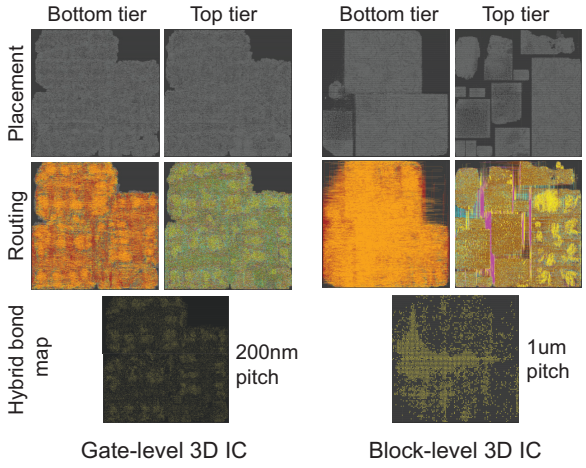

Fig. 9: GDS layouts of gate-level (Pin-3D) vs. block-level (Fence-3D) 3D IC for the JPEG benchmark. Due to feasibility constraints highlighted in Table X, an aggressive 200nm pitch is used for the gate-level 3D IC.

TABLE X: Feasibility of 1um hybrid bond pitch for JPEG among different methods: Pin-3D [16], Pin-in-the-Middle (PITM) [9], and our proposed Fence-3D. The prior works adopt Full-MLS without considering MLS control, whereas Fence-3D leverage Partial-MLS.

	Gate-level 3D	Block-level 3D	
3D design flow	Pin-3D [16]	PITM [9]	Fence-3D
MLS method	Full-MLS	Full-MLS	Partial-MLS
Chip size (μm)	101×101	99.7×99.7	
# $1\mu m$ h-bonds avail.	10,201	9,940	
# hybrid bonds used	59,761	12,307	3,665
$1\mu m$ feasible?	no	no	yes

For all three benchmarks, the No-MLS method effectively reduces the number of hybrid bonds but exhibits relatively low performance. In the JPEG and Rocket designs, this poor performance is attributed to severe routing congestion, as evidenced by high maximum hotspot values on inter-block connection layers (M5 and M6). To verify the impact, detoured nets—those with routed lengths exceeding 1.5× their Manhattan distance—are analyzed, while excluding short local nets with design-specific thresholds of 5, 10, and 15um for AES, JPEG, and Rocket, respectively. The large number of detoured nets observed in the No-MLS serves as evidence that severe congestion increases signal delay, ultimately leading to performance degradation. In contrast, the AES benchmark, which is a cell-dominant benchmark, does not experience full-chip-level congestion under the No-MLS condition. However, we still observe 15 detoured nets due to the absence of MLS-based shortcuts for timing-critical paths.

Compared to the No-MLS method, the Full-MLS approach reduces power and improves performance by enabling flexible routing and mitigating inter-block net congestion. However, this comes at the cost of a substantial increase in hybrid bond usage, which limits the feasibility of 1um pitch. In contrast, the Partial-MLS method selectively applies MLS to highly congested regions. This strategy maintains the total hybrid bond count within the maximum allowable limit for a 1um pitch, while delivering a 4–12% improvement in PDP—comparable to the Full-MLS results. These results demonstrate that Partial-MLS effectively combines the advantages of both No-MLS and Full-MLS, achieving a balance between reduced hybrid bond usage and enhanced power and performance.

TABLE XI: Comparison of metal layer sharing (MLS) methods at the 3nm node: Partial-MLS reduces the hybrid bond count while achieving a power-delay product (PDP) comparable to Full-MLS. Notably, it brings the hybrid bond count within feasible limits for 1um pitch.

	AES (131K cells, 11 blocks)			JPEG (279K cells, 16 blocks)			Rocket (350K, 41 blocks)		
	No-MLS	Full-MLS	Part-MLS	No-MLS	Full-MLS	Part-MLS	No-MLS	Full-MLS	Part-MLS
Chip size	$64\mu m \times 64\mu m$			$99.7\mu m \times 99.7\mu m$			$163\mu m \times 164\mu m$		
# h-bonds avail. ($1\mu m$ pitch)	4,096			9,940			26,732		
# hybrid bonds used	2,165	10,877	3,177	4,485	17,302	5,168	14,098	48,813	15,934
$1\mu m$ pitch feasibility	yes	no	yes	yes	no	yes	yes	no	yes
Wire length (m)	0.316	0.314	0.315	0.773	0.761	0.772	1.126	1.084	1.125
Max. hotspot	11.2	9.9	10.8	170.5	29.3	101.2	108.4	19.3	39.4
# Detoured nets	15	0	2	41	0	12	330	21	105
Power (mW)	75.9	74.1	75.4	296.1	292.1	295.5	195.6	194.4	195.2
Performance	4.34	4.93	4.91	3.23	3.78	3.74	1.72	1.80	1.79
PDP (fJ)	17.5	15.0	15.4	91.7	77.3	79.0	113.7	108.0	109.1
		(-14%)	(-12%)		(-16%)	(-14%)		(-5.0%)	(-4.0%)
Design runtime	8.2h	6.2h	7.2h	10.9h	9.4h	10.7h	21.5h	17.7h	19.0h

TABLE XII: Fence-3D bridges the power and performance gap between gate-level and block-level 3D ICs, while supporting the use of a relaxed 1um hybrid bond pitch and IP reuse. Our reported runtime includes the time required to generate all blocks within the designs.

JPEG benchmark	Gate-3D	Block-3D	
	Pin-3D [16]	PITM [9]	Fence-3D
MLS method	Full	Full	Partial
Block density	N/A	60% uniform	varying
Chip size (μm)	101×101	99.7×99.7	
Wire length (m)	0.53	0.80	0.79
# DRC errors	16	14	13
Hybrid bond pitch	200nm	500nm	$1\mu m$
# hybrid bonds used	59,761	12,307	3,665
Power (mW)	289	302 (+4.7%)	297 (+2.9%)
Performance (GHz)	3.84	3.28 (-15%)	3.70 (-3.6%)
Block design reuse	no	yes	yes
Design runtime	14.5h	19.3h	11.5h

TABLE XIII: Runtime breakdown across different methods for JPEG.

	Gate-3D	Block-3D	
	Pin-3D [16]	PITM [9]	Fence-3D
Block design	-	6.4h	6.4h
3D floorplanning	-	0.4h	0.4h
Mixed-size placement	3h	-	-
Timing closure	11.5h	12.5h	4.7h
Total runtime	14.5h	19.3h	11.5h (-21%)
Total with block reuse	14.5h	12.9h	5.1h (-65%)

E. Closing the Gap between Gate-level and Block-level 3D IC

As previously discussed, 3nm gate-level 3D ICs require a 200nm hybrid bond pitch, which poses a significant gap from current manufacturability. A prior method, Pin-in-the-Middle [9], adopted Full-MLS and resulted in a large number of hybrid bonds, still requiring an aggressive 500nm pitch even with block-level—thus failing to bridge the gap. In contrast, our proposed Partial-MLS method enables a substantial reduction in hybrid bond count. Ultimately, our methodology enables the adoption of a manufacturable 1um pitch in 3nm 3D ICs, as shown in Table X, thereby allowing the benefits of 3D integration to be realized at advanced technology nodes. Note that in Table X, we address hybrid bond overlap by applying the bipartite matching-based legalization technique proposed in [7].

Furthermore, Pin-in-the-Middle [9] lacks post-route optimization, resulting in a 15% performance degradation and a 4.7% increase in power compared to the gate-level 3D IC. In contrast, our final design—with the Fence-3D flow and soft block sizing—successfully narrows the performance and power gap to 3.6% and 2.9%, as shown in Table XII. It demonstrates the effectiveness of our approach in closing the quality gap between gate-level and block-level 3D ICs.

F. Runtime Characterization of 3D Design Flows

Table XIII summarizes the design runtimes for gate-level and block-level 3D ICs. For the gate-level, the flow consists of a pseudo-3D phase—where the design is projected onto a 2D IC—followed by a refinement phase that performs 3D clock and signal routing. In contrast, the block-level flow involves three stages: soft block design, 3D floorplanning, and top-level integration with refinement.

Among the three methods, our proposed Fence-3D combined with Partial-MLS achieves the shortest runtime in the refinement stage, mainly due to the reduced hybrid bonds. This reduction minimizes the exponential overhead of bond legalization, leading to time savings during post-CTS, routing, and post-route optimization. As a result, the proposed method achieves a 21% reduction in total design time compared to the gate-level flow. Furthermore, assuming all blocks are pre-designed, the runtime reduction can reach up to 65%.

VI. Conclusion

In this paper, we presented several methods to design block-level 3D ICs. The proposed Fence-3D approach outperforms the Pin-in-the-Middle flow, achieving up to a 15% improvement in power-delay-product (PDP). Additionally, our ML-based soft block sizing mitigates intra-block congestion, yielding a further 6.6% PDP improvement. Finally, the Partial-MLS method, guided by RUDY map, maintains a low hybrid bond count, thereby enabling the use of a 1um hybrid bond, while achieving up to a 14% improvement in PDP compared to the No-MLS method. These results underscore the significance of our methods in achieving both optimized and manufacturable block-level 3D IC designs for advanced nodes.

Acknowledgment

This work was supported by the Ministry of Trade, Industry Energy of South Korea (1415187652, RS-2023-00234159), the SRC under the JUMP 2.0 CHIMES Center (Task 3136.002), and Samsung Electronics.

REFERENCES

[1] N. E. Bethur, A. Agnesina, M. Brunion, A. Garcia-Ortiz, F. Catthoor, D. Milojevic, M. Komalan, M. Cavalcante, S. Riedel, L. Benini, et al. Hier-3D: A Methodology for Physical Hierarchy Exploration of 3-D ICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 43(7):1957–1970, 2023.

[2] Y.-W. Chang, Z.-W. Jiang, and T.-C. Chen. Essential Issues in Analytical Placement Algorithms. *IPSJ Transactions on System and LSI Design Methodology*, 2:145–166, 2009.

[3] Y.-J. Chen, Y.-S. Chen, W.-C. Tseng, C.-Y. Chiang, Y.-H. Lo, and Y.-W. Chang. Late Breaking Results: Analytical Placement for 3D ICs with Multiple Manufacturing Technologies. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–2. ACM/IEEE, 2023.

[4] S.-A. Chew, S. Iacovo, F. Fordor, S. Dewilde, K. Devriendt, J. De Vos, A. Miller, G. Beyer, and E. Beyne. 700nm pitch Cu/SiCN wafer-to-wafer hybrid bonding. In *2022 IEEE 24th Electronics Packaging Technology Conference (EPTC)*, pages 334–337. IEEE, 2022.

[5] J. Cong, J. Wei, and Y. Zhang. A Thermal-Driven Floorplanning Algorithm for 3D ICs. In *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.*, pages 306–313. IEEE, 2004.

[6] J. Hu, P. Vanna-iampikul, Z. Zhuang, T.-Y. Ho, and S. K. Lim. GNN-MLS: Signal Routing in Mixed-Node 3D ICs through GNN-Assisted Metal Layer Sharing. In *2025 62th ACM/IEEE Design Automation Conference (DAC)*. ACM/IEEE, 2025.

[7] Y.-H. Huang, S. Pentapati, A. Agnesina, M. Brunion, and S. K. Lim. On Legalization of Die Bonding Bumps and Pads for 3D ICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[8] D. Kim, H. Kwon, S.-Y. Lee, S. Kim, M. Woo, and S. Kang. Machine Learning Framework for Early Routability Prediction with Artificial Netlist Generator. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1809–1814. IEEE, 2021.

[9] J. Kim, B. W. Ku, J. Yoon, and S. K. Lim. An Effective Block Pin Assignment Approach for Block-Level Monolithic 3-D ICs. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, 7(1):26–34, 2021.

[10] T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907*, 2016.

[11] B. W. Ku, K. Chang, and S. K. Lim. Compact-2D: A Physical Design Methodology to Build Commercial-Quality Face-to-fFace-Bonded 3D ICs. In *Proceedings of the 2018 International Symposium on Physical Design*, pages 90–97, 2018.

[12] J.-M. Lin and Y.-W. Chang. TCG-S: Orthogonal Coupling of P*-Admissible Representations for General Floorplans. In *Proceedings of the 39th Annual Design Automation Conference*, pages 842–847, 2002.

[13] Y.-C. Lu, W.-T. Chan, D. Guo, S. Kundu, V. Khandelwal, and S. K. Lim. RL-CCD: Concurrent Clock and Data Optimization using Attention-Based Self-Supervised Reinforcement Learning. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. ACM/IEEE, 2023.

[14] Y.-C. Lu, K. Kunal, G. Pradipta, R. Liang, R. Gandikota, and H. Ren. LEGO-Size: LLM-Enhanced GPU-Optimized Signoff-Accurate Differentiable VLSI Gate Sizing in Advanced Nodes. In *Proceedings of the 2025 International Symposium on Physical Design*, pages 152–162, 2025.

[15] S. Pentapati and S. K. Lim. Metal Layer Sharing: A Routing Optimization Technique for Monolithic 3D ICs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 30(9):1355–1367, 2022.

[16] 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 *Proceedings of the 39th International Conference on Computer-Aided Design*, pages 1–9, 2020.

[17] X. Tang and D. Wong. FAST-SP: A Fast Algorithm for Block Placement based on Sequence Pair. In *Proceedings of the 2001 Asia and South Pacific design automation conference*, pages 521–526, 2001.

[18] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, et al. Graph Attention Networks. *stat*, 1050(20):10–48550, 2017.

[19] E. Wong and S. K. Lim. 3D Floorplanning with Thermal Vias. In *Proceedings of the Design Automation & Test in Europe Conference*, volume 1, pages 1–6. IEEE, 2006.

[20] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim. Graph Transformer Networks. *Advances in neural information processing systems*, 32, 2019.