DCO-3D: Differentiable Congestion Optimization in 3D ICs

Hao-Hsiang Hsiao[1], Yi-Chen Lu[2], Pruek Vanna-iampikul[3], Anthony Agnesina[2], Rongjian Liang[2], Yuan-Hsiang Lu[1], Haoxing Ren[2], and Sung Kyu Lim[1]
[1]School of ECE, Georgia Institute of Technology, Atlanta, GA
[2]NVIDIA, Santa Clara, CA, USA; [3]Deparment of Electrical Engineering, Burapha University, Chonburi, Thailand;
{thsiao, yuan-hsiang.lu, limsk}@gatech.edu; {yilu, aagnesina, rliang, haoxingr}@nvidia.com;

***Abstract*—State-of-the-art 3D IC flows fail to consider 3D congestion during earlier stages, leading to excessive use of end-of-flow ECO resources for routability correction that severely degrades full-chip Power, Performance, and Area metrics. We present DCO-3D, a Machine Learning-based routability-aware 3D PD flow that performs early post-route congestion prediction using Siamese Networks and resolves the predicted hotspots using a fully differentiable 3D cell spreading with Graph Neural Network. On 6 industrial designs in a commercial 3nm node, DCO-3D improves Pin-3D, the known best Pin-3D flow, by up to 47.2% in overflow, 86.2% in TNS and 5.1% in power at signoff.**

I. Introduction

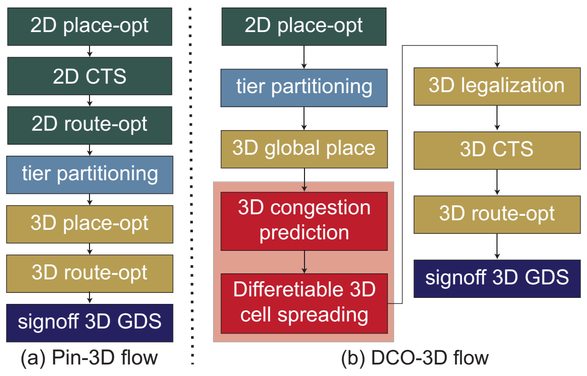
State-of-the-art (SOTA) 3D Physical Design (PD) flows leverage 2D commercial Place and Route (P&R) tools to build signoff-quality 3D Integrated Circuits (ICs), a methodology commonly referred to as the "pseudo" approach [1]. Despite significant advances in improving the Power, Performance, and Area (PPA) envelope, the challenge of routability in 3D ICs remains a critical and underexplored issue, posing a bottleneck to further advancements in 3D integration.

Decades of research in 2D PD have underscored the critical role of routability optimization during the placement stage in achieving superior full-chip PPA metrics [2], [3]. However, this foundational design principle has yet to be fully adopted in the context of 3D ICs. The advent of Machine Learning (ML)-based data-driven approaches has driven the development of various methodologies [4], [5], [6], [7], [8], [9] aimed at predicting routability in early PD stages, enabling proactive optimization. However, these advancements are limited to 2D ICs and cannot be seamlessly extended to 3D ICs due to the unique and inherent complexity of 3D designs.

To address these limitations, we propose DCO-3D, the first ML-driven, routability-aware 3D PD flow that incorporates 3D congestion prediction using customized Siamese networks [10]. Building on this prediction, we introduce a fully differentiable 3D congestion optimization framework that minimizes congestion while preserving placement quality. Unlike prior 2D approaches, our framework enables cell spreading across all three dimensions—x, y, and z—facilitating congestion redistribution not only horizontally and vertically but also across tiers. This cross-tier capability effectively utilizes 3D resources to resolve congestion hotspots that are intractable in conventional 2D layouts.

DCO-3D can seamlessly integrate into any existing 3D PD flow. In this work, we specifically compare it against Pin-3D [11], the widely recognized SOTA 3D design flow, to highlight its advantages. Figure 1 illustrates how our DCO-3D methodology (highlighted in red) integrates with the Pin-3D flow.

DCO-3D takes a 3D global placement as input and directly generates cell spreading decisions in TCL constraints for the commercial P&R tool, *Synopsys ICC2*. Notably, DCO-3D introduces no additional PD optimization steps beyond those already present in the Pin-3D flow. Instead, it provides supplemental TCL and Python scripts to guide cell spreading decisions for congestion optimization within the existing tool. Experimental results demonstrate that this early optimization yields significant full-chip PPA improvements by the end of the flow. This work represents the *first* routability-aware 3D PD flow, offering accurate and efficient congestion optimization during the early design stages. Our main contributions are as follows:



🟩 : pseudo-2D stages 🟨 : concurrent multi-die opt. 🟥 : 3D concurrent DCO opt.

Fig. 1: SOTA 3D PD flow Pin-3D [11] vs. our ML-driven routability-aware 3D flow DCO-3D, which introduces additional differentiable 3D cell spreading for early and precise congestion optimization.

- We propose DCO-3D, the first commercial-quality 3D PD flow that performs congestion optimization for 3D ICs.
- We introduce the first congestion optimization framework that leverages the z-dimension, allowing cells to dynamically move between tiers to optimize congestion.
- We develop a fully differentiable, multi-objective cell spreading methodology using GNN, facilitating direct gradient-based optimization.
- We design a customized Siamese network tailored for concurrent multi-die 3D congestion prediction. Existing 2D routability prediction methods [4], [5], [6], [7], [8], [9] are inadequate for addressing such 3D concurrent challenges.
- We achieve significant 3D full-chip PPA improvements over the SOTA 3D PD flow Pin-3D [11], demonstrated on six industrial designs in a commercial $3nm$ technology node.

II. Preliminaries

A. State-of-the-art 3D Flow

Pseudo-3D [1], unlike academic true-3D approaches [12], [13], [14], [15], [16], utilizes commercial tools to construct 3D ICs, ensuring commercial PPA quality and manufacturing readiness in GDS format. The Pseudo-3D flow [17] strategically leverages commercial tools to determine the optimal (x, y) locations of standard cells while assigning z-coordinates through tier assignments. In this work, we adopt the SOTA Pin-3D [11] flow to directly optimize 3D placement for improved routability. As illustrated in Figure 1, the process encompasses placement, Clock Tree Synthesis (CTS), post-CTS optimization, routing, and timing closure, all within a 3D context.

B. Early Congestion Estimation

1) RUDY (Rectangular Uniform wire DensitY)

RUDY[18] is routing demand estimation based solely on the pin locations. The RUDY of a net e can be computed as follows: For a net with bounding box coordinates $\{x_e^{\mathrm{l}}, y_e^{\mathrm{l}}, x_e^h, y_e^h\}$, its RUDY at any location (x, y) is proportional to the wire area divided by the bounding box area:

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

where $w_e = x_e^{\mathrm{h}} - x_e^{\mathrm{l}}$ and $h_e = y_e^{\mathrm{h}} - y_e^{\mathrm{l}}$, and μ_e is an indicator function that checks whether (x, y) lies within the bounding box of net e. Given the design is divided into multiple tiles, the RUDY at tile (m, n) is:

$$RUDY_{(m,n)} = \sum_{e \in N} \left(\frac{1}{w_e} + \frac{1}{h_e}\right) \frac{\text{Area}_{tile \cap bbox}}{\text{Area}_{tile}} \quad (2)$$

2) PinRUDY

PinRUDY, an extension of RUDY for estimating pin density, is defined as follows:

$$\text{PinRUDY}(m,n) = \sum_{pin \in tile_{mn}} \left(\frac{1}{w_e} + \frac{1}{h_e}\right), \quad pin \in e \quad (3)$$

C. Overview of Our Approach

This study addresses two critical challenges in 3D IC design: (1) predicting post-3D routing congestion maps from a given 3D global placement ("images-to-images" prediction), and (2) refining placements based on these predictions to mitigate congestion hotspots while preserving signoff-quality QoR.

Our methodology consists of two main stages:

1) 3D Congestion Prediction: Predict post-route congestion maps to identify potential hotspots.
2) 3D Congestion Optimization: Utilize these predictions to perform 3D cell spreading and alleviate congestion.

III. ML-based Routability Prediction

A. Dataset Construction

The proposed prediction framework relies on a prebuilt dataset for supervised learning. Using our in-house 3nm process design kit (PDK), we synthesize RTL into gate-level netlists with Synopsys Design Compiler, followed by Pseudo-3D P&R using *Synopsys IC Compiler II (ICC2)*. For each netlist, 300 diverse 3D placement layouts are generated by sampling placement parameters listed in Table I, serving as training input. Ground truth congestion maps are obtained by completing 3D CTS and routing for each layout.

B. Data Engineering

In this subsection, we describe the input features of the prediction model, the training labels, and the data processing methods.

1) Input Feature Maps

During global routing, the chip layout is divided into routing bins, referred to as GCells. Our approach partitions each die in a 3D placement based on these GCell coordinates to generate feature maps that aid in predicting the post-3D routing layout. Each pixel in these maps encapsulates specific local information corresponding to its respective bin area. We designed the following feature maps specifically for 3D ICs:

- Cell Density: The ratio of cell area within a bin to the bin's area.
- Pin Density: The number of pins per unit area
- 2D RUDY: Routing demand for 2D nets (nets with all pins on a single die), as described in Section II-B.

TABLE I: 3D Placement Parameters used for constructing placement dataset

Placement Parameter	type	value
coarse.pin_density_aware	bool	"false", "true"
coarse.target_routing_density	float	[0, 1]
coarse.adv_node_cong_max_util	float	[0, 1]
coarse.congestion_driven_max_util	float	[0, 1]
coarse.cong_restruct_effort	enum	[0, 4]
coarse.cong_restruct_iterations	int	[0, 10]
coarse.enhanced_low_power_effort	enum	[0, 4]
coarse.low_power_placement	bool	"false", "true"
coarse.max_density	float	[0, 1]
legalize.displacement_threshold	int	[0, 10]
initial_place.two_pass	bool	"false", "true"
initial_drc.global_route_based	bool	"false", "true"
flow.enable_ccd	bool	"false", "true"
initial_place.effort	enum	[0, 2]
final_place.effort	enum	[0, 2]
flow.enable_irap	bool	"false", "true"

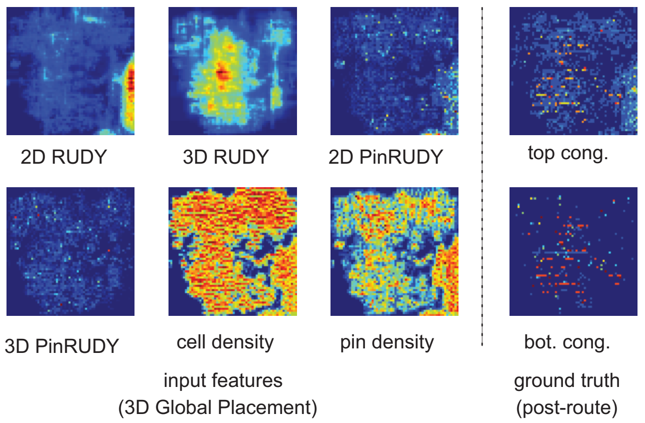


Fig. 2: Visualization of input features and ground truth: "2D" features represent nets with all pins located on the same die, whereas "3D" features capture nets with pins spanning across different dies.

- 3D RUDY: Routing demand for 3D nets (nets connecting pins across multiple dies), scaled by **0.5** to account for additional 3D routing resources.
- 2D PinRUDY: Pin-based routing demand II-B for 2D nets.
- 3D PinRUDY: Pin-based routing demand II-B for 3D nets.
- Macro Blockage: Area occupied by macros.

2) Training Label Maps

The training labels for each data point are congestion maps for both the top and bottom dies, as the objective is to predict these maps concurrently. Each congestion map is divided into bins corresponding to GCells, with each bin assigned a routing overflow value derived from the congestion report generated by the tool. A visualization of these maps from a selected data sample is presented in Figure 2.

3) Pre- and Post-Processing of Maps

The overall data processing pipeline is illustrated in Figure 3. For each design in the dataset, the placement layout is divided into bins based on GCell dimensions to compute the feature maps. Similarly, the post-route layout is divided into GCell bins to generate the label maps. To handle varying GCell grid sizes across different designs, both feature and label maps are resized to a fixed dimension of $H \times W$, where H and W denote the height and width required by the convolutional neural network (CNN). In this study, we set $H = W = 224$. During preprocessing, nearest neighbor interpolation is used to resize each map, preserving the original pixel magnitudes during both upscaling and downscaling. This ensures accurate recovery of the original map after transformation. Post-training, the output maps are

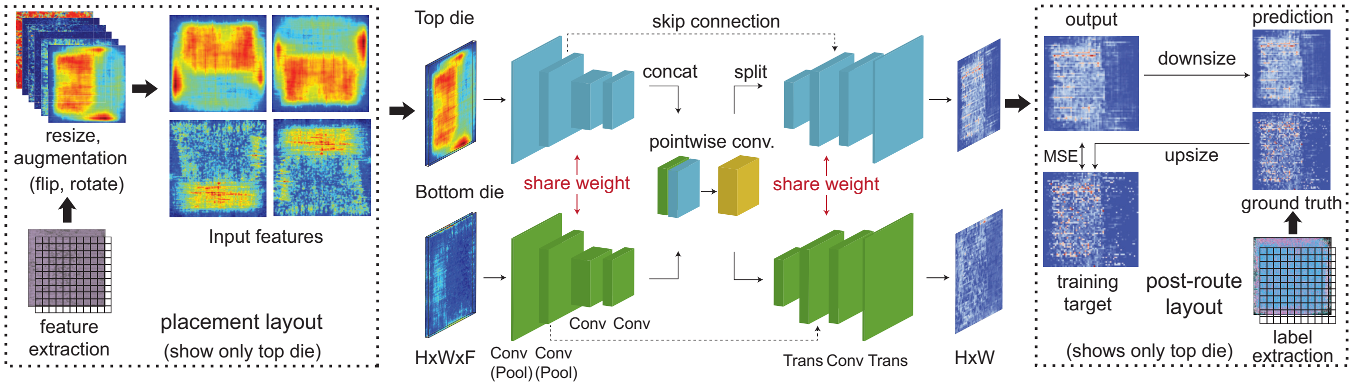


Fig. 3: Our 3D images-to-images Siamese UNet employs a shared encoder-decoder architecture for the top and bottom dies in the F2F 3D IC design. A pointwise communication convolutional layer enables efficient information exchange between the dies, enabling concurrent post-route congestion hotspot prediction in a fully 3D manner.

Algorithm 1 Congestion Prediction

```
Input: Dataset $\mathcal{D} = \{(F_0^i, F_1^i, C_0^i, C_1^i)\}$ with features $F_0^i, F_1^i \in \mathbb{R}^{h\times w\times 7}$ and targets $C_0^i, C_1^i \in \mathbb{R}^{h\times w}$
Output: Predicted congestion maps $\{\hat{C}_0^i, \hat{C}_1^i \in \mathbb{R}^{h\times w}\}$
1: Resize features: $F_d^i \in \mathbb{R}^{h\times w\times 7} \to \mathbb{R}^{H\times W\times 7} \quad \forall d \in \{0,1\}$
2: Resize targets: $C_d^i \in \mathbb{R}^{h\times w} \to \mathbb{R}^{H\times W} \quad \forall d \in \{0,1\}$
3: Initialize SiaUNet with parameters $\theta$
4: while not converged do
5:     $\{\hat{C}_0^i, \hat{C}_1^i\} \leftarrow \text{SiaUNet}(F_0^i, F_1^i; \theta)$
6:     Compute loss: $L = \frac{1}{2}\sum_{d\in\{0,1\}}\sqrt{\frac{1}{HW}\|\hat{C}_d^i - C_d^i\|_F^2}$
7:     Update parameters: $\theta \leftarrow \theta - \eta\nabla_\theta L$
8: end while
```

resized back to their original dimensions for inference, maintaining consistency with the initial grid structure and ensuring compatibility with subsequent design analyses. To improve the model's robustness to layout orientation, we applied random transformations—-rotations by 0°, 90°, 180°, and 270°, as well as horizontal and vertical flipping—-during training. This data augmentation strategy increases the diversity of training samples by a factor of eight and enhances the model's generalizability to various layout orientations.

C. Overview of Our Prediction Model

Our customized 3D prediction model architecture, illustrated in Figure 3, utilizes feature maps from the 3D global placements of both dies as inputs to predict their post-route congestion maps. The model is built on the UNet architecture, which is well-suited for image-to-image prediction tasks. It features a downsampling encoder and an upsampling decoder: the encoder progressively reduces the spatial dimensions of the input images while increasing the number of feature channels to capture contextual features, and the decoder restores the spatial dimensions by reducing the feature channels through transposed convolutions, ultimately generating the predicted congestion maps. To retain finer details and mitigate information loss, skip connections are incorporated between the encoder and decoder.

Given the interchangeable nature of the top and bottom dies in our face-to-face bonded design, we propose a customized Siamese 3D UNet architecture. This architecture employs shared weights for the encoders and decoders of both dies, ensuring identical feature extraction and reconstruction processes. To capture inter-die dependencies, we introduce a communication layer between the encoder and decoder. This layer merges the encoder outputs from both dies, processes them through pointwise convolution to enable inter-die channel communication, and subsequently splits the processed

Algorithm 2 Differentiable Congestion Optimization (DCO)

```
Input: Trained congestion predictor SiaUNet*, initial 3D placement $\mathbf{P} = [\mathbf{x}, \mathbf{y}, \mathbf{z}] \in \mathbb{R}^{N\times 3}$, netlist graph $\mathbf{G}$, cell attributes $\mathbf{h}$
Output: Optimized 3D placement $\mathbf{P}^*$
1: Initialize GNN parameters $\theta$
2: while iter < max_iter and not converged do
3:     $\mathbf{P}' \leftarrow \text{GNN}_\theta(\mathbf{G}, \mathbf{h})$                ▷ Updated 3D placement
4:     $L_{\text{disp.}}, L_{\text{ovlp.}}, L_{\text{cutsize}} \leftarrow$ Compute losses$(\mathbf{P}, \mathbf{P}')$
5:     $F_d \leftarrow$ Generate features$(\mathbf{P}') \quad \forall d \in \{0,1\}$
6:     $C'_d \leftarrow \text{SiaUNet}^*(F_d) \quad \forall d \in \{0,1\}$       ▷ Congestion prediction
7:     $L_{\text{cong.}} \leftarrow$ Compute congestion loss$(C'_d)$
8:     $L_{\text{total}} = \alpha L_{\text{disp.}} + \beta L_{\text{ovlp.}} + \gamma L_{\text{cutsize}} + \delta L_{\text{cong.}}$
9:     $\theta^{i+1} \leftarrow \theta^i - \eta \cdot \frac{\partial L_{\text{total}}}{\partial \theta}$                ▷ Gradient update
10: end while
11: return $\mathbf{P}^* \leftarrow \text{GNN}_{\theta^*}(\mathbf{G}, \mathbf{h})$         ▷ Optimized 3D placement
```

features back into two streams for decoding. Overall, our model is an images-to-images framework that takes the feature map tensors of both dies, $\mathbf{F}_0$, $\mathbf{F}_1 \in \mathbb{R}^{H\times W\times 7}$, as input, and outputs the corresponding post-route congestion maps, $\mathbf{C}_0$, $\mathbf{C}_1 \in \mathbb{R}^{H\times W}$; we set the dimensions to $H = W = 224$.

D. Training Our Prediction Model

The complete training procedure is described in Algorithm 1. The training of our prediction model is guided by the sum of the root mean squared Frobenius losses between the predicted congestion maps, $\hat{\mathbf{C}} \in \mathbb{R}^{H\times W}$, and the ground truth label maps, $\mathbf{C} \in \mathbb{R}^{H\times W}$ for both dies. This objective function is expressed as follows:

$$L = \frac{1}{2}\sum_{d\in\{0,1\}}\sqrt{\frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}\left(\hat{\mathbf{C}}_d(i,j) - \mathbf{C}_d(i,j)\right)^2} \tag{4}$$

IV. Differentiable Congestion Optimization

Our optimization framework is fully differentiable and three-dimensional, allowing for direct, gradient-based refinement of cell placement to enhance congestion metrics and placement quality.

The optimization consists of the following core steps:

1) **GNN-based 3D Cell Spreading:** A GNN predicts updated cell locations to mitigate congestion and enhance placement quality.
2) **Differentiable Loss Functions:** Custom loss functions—including congestion, displacement, overlap, and mincut—guide the GNN during training.
3) **End-to-End Optimization:** Gradients are backpropagated to iteratively update the GNN using gradient descent.

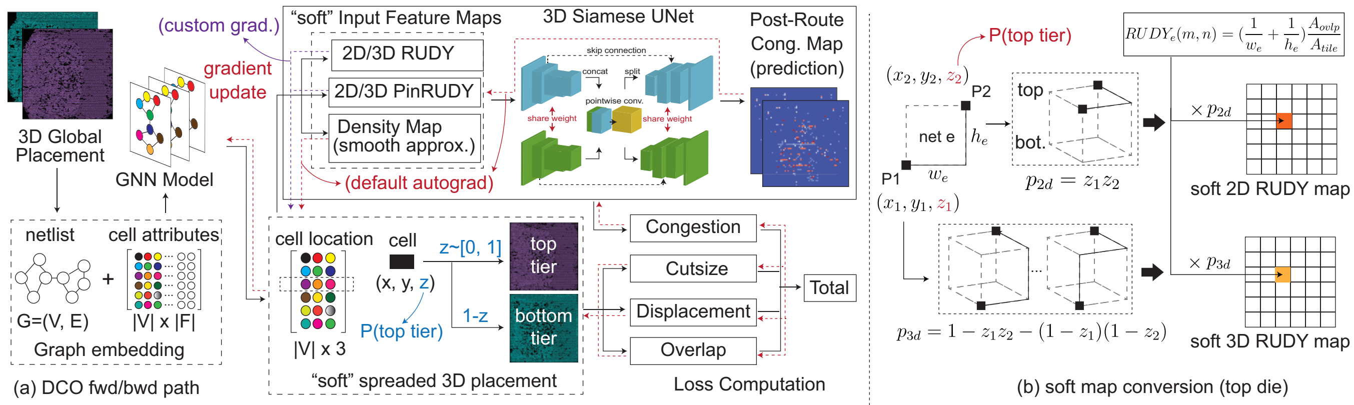


Fig. 4: Differentiable 3D Congestion Optimization Flow: The process starts with a 3D global placement result. The GNN predicts new cell locations as soft probabilistic assignments, partially contributing to both top and bottom tiers. These locations are converted into "soft" feature maps and processed by a 3D Siamese U-Net to predict post-route congestion. The predicted congestion guides GNN optimization with other objectives. A custom backward function handles non-differentiable components, enabling gradient-based optimization of cell locations to achieve design goals.

TABLE II: Initial handcrafted features of each node in the netlist graph.

features	**descriptions**
wst slack	worst slack of cell
wst output slew	maximum transition of output pin
wst input slew	maximum transition of input pin
drv net power	switching power of driving net
int power	cell internal power
leakage	cell leakage power
width	cell width
height	cell height

The optimization iterates until convergence, as outlined in Algorithm 2. Starting from a 3D global placement, the GNN predicts updated cell locations as soft probabilistic assignments, allowing cells to contribute to both tiers. These locations are transformed into feature maps and passed to the 3D Siamese U-Net to predict post-route congestion. The predicted congestion, along with other objectives, guides GNN optimization. To handle non-differentiable components, a custom backward function enables direct gradient-based refinement of cell locations for multi-objective optimization.

A. Differentiable 3D Cell Spreading

Our cell spreading approach jointly optimizes routability and placement quality using differentiable loss functions that balance congestion, overlap, displacement, and cutsize. Their differentiability enables gradient-based optimization of cell locations. Unlike 2D methods limited to horizontal and vertical spreading, our 3D solution redistributes cells across dies. This added flexibility in the z-direction effectively resolves congestion hotspots that are unsolvable in traditional 2D layouts. To efficiently optimize cell placement, we avoid learning independent (x, y, z) coordinates per cell, which would scale poorly for large netlists with millions of parameters. Instead, we utilize a GNN consisting of three Graph Convolutional Network (GCN) layers with shared weights across all cells. This approach leverages cell connectivity and intrinsic characteristics (Table II), allowing globally informed updates and stable convergence. The GNN predicts (x, y, z), where x and y are continuous 2D positions, and $z \in [0, 1]$ represents the probability of assigning a cell to the top die (z) or bottom die ($1 - z$). This probabilistic z promotes differentiable optimization across tiers, allowing each cell to influence both dies before final hard assignment via $z = \mathbb{1}z \geq 0.5$. During training, z guides soft tier assignments in feature and density maps. For RUDY maps (Fig. 4b), the 2D contribution is weighted by $\prod p \in ez_p$ (top) or $\prod_{p\in e}(1-z_p)$ (bottom), while the 3D contribution is weighted by $1 - \prod_{p\in e} z_p - \prod_{p\in e}(1 - z_p)$. These maps inform loss computations for gradient-based optimization.

B. Congestion Loss

After the GNN generates new cell locations, the input feature maps (Sections III-B1 and IV-A) are processed by the model (Figure 3) to predict congestion maps $\mathbf{C} \in \mathbb{R}^{H\times W\times 2}$ for the top and bottom dies. The congestion penalty is calculated using Eq. 4. The gradient with respect to the GNN parameters θ is computed via the chain rule:

$$\frac{\partial L}{\partial \theta} = \sum_{d\in[\text{top},\text{bottom}]} \frac{\partial L}{\partial \mathbf{C}_d} \cdot \frac{\partial \mathbf{C}_d}{\partial \mathbf{F}_d} \cdot \frac{\partial \mathbf{F}_d}{\partial \mathbf{X}_{\text{cells}}} \cdot \frac{\partial \mathbf{X}_{\text{cells}}}{\partial \theta} \tag{5}$$

$\partial L/\partial \mathbf{C}_d$ measures how the loss changes with respect to the predicted congestion map $\mathbf{C}_d$. The term $\partial \mathbf{C}_d/\partial \mathbf{F}_d$ represents the sensitivity of congestion predictions to changes in the feature maps $\mathbf{F}_d$, while $\partial \mathbf{F}_d/\partial \mathbf{X}_{\text{cells}}$ describes how feature maps are influenced by cell positions. Since $\partial \mathbf{F}_d/\partial \mathbf{X}_{\text{cells}}$ is non-differentiable at grid boundaries, we implement a custom PyTorch backward function that approximates gradients using subgradients for non-differentiable components while leveraging PyTorch's native autograd for smooth terms. For each tile (m, n), we compute the gradient of RUDY values with respect to cell positions by summing contributions from all crossing nets. The sensitivity of a net e's RUDY value in tile (m, n) to the x-position of cell i $\partial RUDY_e(m, n)/\partial x_i$ is given by:

$$\mu_e(m, n) \left[\frac{(w_e - w')h'}{w_e^2} + \frac{h'}{h_e}\right] \frac{1}{A_{\text{GCell}}}(\delta_{ih} - \delta_{il}) \tag{6}$$

where w', h' are the dimensions of the net-tile overlap, A_{GCell} is the tile area, and $\mu_e(m, n)$ indicates whether net e crosses tile (m, n). The term $(\delta_{ih} - \delta_{il})$ represents changes in the net's bounding box due to cell i's movement. Here, δ is the Kronecker delta function, which is nonzero only when cell i contains the leftmost ($\delta_{il} = 1$) or the rightmost ($\delta_{ih} = 1$) pin of the net.

C. Cutsize Loss

While 3D spreading allows cell movements between dies to alleviate congestion hotspots, it is essential to minimize the cut size—the number of inter-die connections—as excessive cuts can increase fabrication costs and degrade circuit performance. To address this, we incorporate a min-cut loss into our optimization:

$$L_{cut}(T, B) = \frac{\text{cut}(T, B)}{\text{deg}(T)} + \frac{\text{cut}(T, B)}{\text{deg}(B)} \tag{7}$$

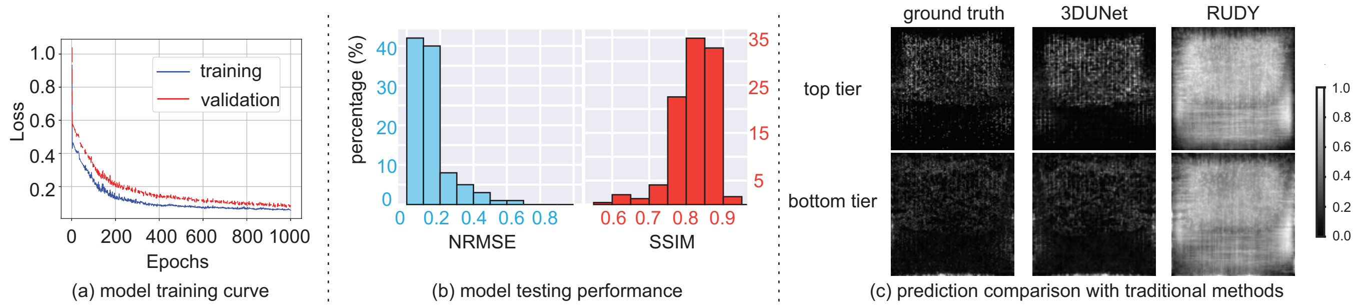


Fig. 5: Evaluation of prediction results: (a) Training and testing loss curves, (b) X-axis denotes absolute NRMSE/SSIM values; y-axis denotes percentage of testing samples. Both metrics fall in desired regions. (c) Comparison of predicted, traditional, and ground-truth congestion maps.

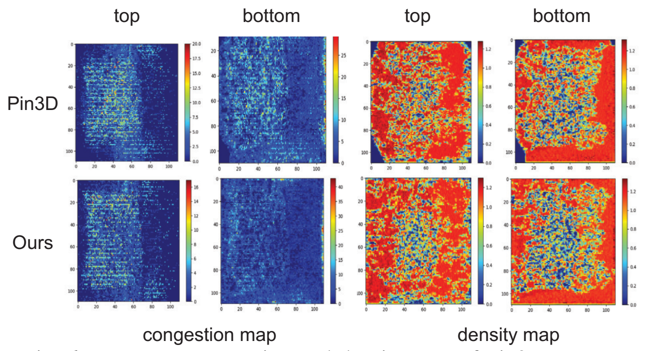


Fig. 6: Post-route congestion and density map of Pin3D vs. ours

Here, cut(T, B) counts the edges connecting cells between the top die T and the bottom die B, while deg(T) and deg(B) represent the total connections within each die. This normalized formulation balances inter-die and intra-die connectivity, ensuring a manageable number of inter-die connections for fabrication.

D. Overlap Loss

To address overlap in cell spreading efficiently, we use a density loss metric to promote spatial separation. Since density function $D_b(x, y)$ are typically non-smooth and non-differentiable, we apply smoothing using bell-shaped potential functions for both the x and y directions. These functions are piecewise-defined based on the center-to-center distance d_x or d_y between the block and the bin:

$$p_x(b, v) = \begin{cases} b\left(1 - ad_x^2\right) & \text{if } 0 \le d_x \le w_b + \frac{w_v}{2}, \\ b\left(d_x - \frac{w_v}{2} - 2w_b\right)^2 & \text{if } 2w_b + \frac{w_v}{2} \le d_x \le \frac{w_b + 2w_v}{2}, \\ 0 & \text{otherwise}, \end{cases} \tag{8}$$

where w_b and w_v are the block and bin width, and a and b are smoothing parameter:

$$a = \frac{4}{(w_v + 2w_b)(w_v + 4w_b)}, \quad b = \frac{2}{w_b(w_v + 4w_b)}. \tag{9}$$

Our objective is the density function $\hat{D}_b(x, y)$:

$$\hat{D}_b(x, y) = \sum_{v \in V} c_v p_x(b, v) p_y(b, v). \tag{10}$$

E. Displacement Loss

To preserve the quality of the optimized 2D positions in initial 3D global placement, we constructed a displacement loss function to constrain cell movement. Displacement loss minimizes the deviation of cells' current positions from their original locations, preserving the quality of the 3D global placement. The loss is defined as:

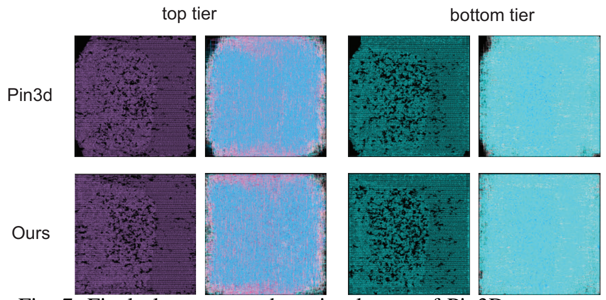


Fig. 7: Final placement and routing layout of Pin3D vs. ours.

$$L_{disp.} = \sum_i (x_i - x_i^o)^2 + (y_i - y_i^o)^2 \tag{11}$$

V. Experimental Results

We perform thorough experiments by comparing our DCO-3D flow against Pin-3D [11], the state-of-the-art 3D design flow and its improved variations on 6 industrial designs, including RocketCore, LDPC, AES, ECG, DMA, and VGA. All the benchmarks are synthesized under a commercial 3nm technology node using *Synopsys Design Compiler* with 1μm pitch face-to-face 3D hybrid bonding. We leverage *Synopsys ICC2* to perform pseudo-2D and 3D P&R. The Siamese network and the differentiable GNN-based cell spreader, are implemented in Python using PyTorch and PyTorch Geometric.

A. Prediction Results

To train the proposed Siamese architecture, illustrated in Figure 3, we construct our dataset as described in Section III-A, reserving 20% as the testing set. To evaluate the spatial accuracy and visual similarity of the model's map predictions, we use two metrics for 2D grid signals: normalized root mean square error (NRMSE) and structural similarity index (SSIM). NRMSE measures the discrepancy between predictions and observed values, with lower values indicating higher accuracy. For image prediction tasks, an NRMSE below 0.2 signifies close alignment with actual images. SSIM quantifies structural similarity between images, ranging from -1 to 1, where 1 denotes identical images. SSIM values above 0.7 are sufficient.

Figure 5 presents our Siamese UNet-based model's prediction results, including training curves, metrics, and visualizations. As shown in Figure 5 (b), over 85% of test samples achieve NRMSE below 0.2 and SSIM above 0.8, indicating high prediction accuracy. Figure 5 (c) compares our model with the RUDY estimator on a test sample (AES design), with pixel values normalized to $[0, 1]$ for fairness. While RUDY poorly correlates with the ground truth in 3D ICs, our 3D IC-customized Siamese UNet shows far higher similarity, validating its effectiveness as a reliable proxy for post-route routability optimization during 3D global placement.

TABLE III: Detailed comparisons of optimization results between state-of-the-art 3D flow Pin-3D [11], Pin-3D with *Synopsys ICC2* congestion-focus optimization (Pin-3D + Cong.), Pin-3D with Bayesian Optimization (Pin-3D + BO), and the proposed DCO-3D flow over 6 industrial benchmarks in a commercial $3nm$ technology node. The best metric in each column is colored in red. Note that we use the exact same *Synopsys ICC2* seed across all experiments to completely remove run-to-run non-determinism.

evaluation	after 3D placement optimization				after signoff optimization (end-of-flow)			
stage	overflow	ovf. gcell%	H ovf.	V ovf.	setup wns (ps)	setup tns (ps)	total power (mW)	WL (μm)
DMA (#cells: 13K, #nets: 14K, #IO: 961)								
Pin3D [11]	3389	26.17	1914	1475	-24.45	-5277	11.3	25572.14
Pin3D + Cong.	2352	23.62	693	1659	-42.67	-6439	11.2	26705.18
Pin3D + BO	2214	21.96	573	1641	-58.87	-17503	11.2	26387.22
DCO-3D (ours)	2018 (-40.45%)	20.15	586	1432	-22.05 (-9.82%)	-4443 (-15.79%)	10.9 (-3.54%)	25076.70
AES (#cells: 114K, #nets: 114K, #IO: 390)								
Pin3D [11]	34158	32.11	18149	16009	-35.64	-10877	75.1	243430.25
Pin3D + Cong.	21018	23.13	5617	15401	-46.27	-11714	74.4	246183.06
Pin3D + BO	19362	22.07	3872	15490	-52.28	-12049	74.1	252131.34
DCO-3D (ours)	18026 (-47.23%)	21.99	2498	15528	-27.15 (-23.82%)	-7928 (-27.11%)	72.6 (-3.33%)	233401.63
ECG (#cells: 83K, #nets: 84K, #IO: 1.7K)								
Pin3D [11]	15083	19.32	548	14535	-54.96	-41163	78.0	200362.67
Pin3D + Cong.	14137	18.49	559	13578	-75.25	-90069	77.9	199021.49
Pin3D + BO	13929	18.27	607	13322	-78.28	-47713	75.9	207186.17
DCO-3D (ours)	13686 (-9.26%)	17.06	1520	12166	-41.47 (-24.55%)	-12394 (-69.89%)	75.9 (-2.69%)	198998.32
LDPC (#cells: 39K, #nets: 41K, #IO: 4.1K)								
Pin3D [11]	17666	38.90	3318	13709	-154	-145389	71.3	211438.43
Pin3D + Cong.	13637	33.81	1335	12302	-160.32	-167845	71.1	214667.37
Pin3D + BO	13091	31.43	1668	11423	-169.13	-193429	71.0	220390.83
DCO-3D (ours)	12511 (-26.52%)	29.77	2685	9826	-75.68 (-50.86%)	-69314 (-52.33%)	67.6 (-5.19%)	209304.86
VGA (#cells: 52K, #nets: 52K, #IO: 184)								
Pin3D [11]	38523	34.38	2766	35757	-20.16	-114.77	17.1	124202.71
Pin3D + Cong.	34005	38.69	929	33076	-150.53	-4505	16.9	138850.31
Pin3D + BO	33969	38.99	700	33269	-227.41	-12170	16.7	147160.69
DCO-3D (ours)	32578 (-15.43%)	37.42	905	31673	-10.3 (-48.91%)	-25.14 (-78.10%)	16.5 (-3.51%)	123712.41
Rocket (#cells: 120K, #nets: 120K, #IO: 379)								
Pin3D [11]	27742	7.00	818	26924	-308.95	-20321	31.2	345382.06
Pin3D + Cong.	22958	5.81	627	22331	-605.58	-59599	30.6	350021.31
Pin3D + BO	22286	5.73	613	21673	-779.8	-183225	31.2	363796.28
DCO-3D (ours)	17963 (-35.25%)	4.49	1460	16503	-147.33 (-52.31%)	-2800 (-86.22%)	29.7 (-4.81%)	328475.40

B. Optimization Results

Figure 6 and 7 show the post-route layout, congestion maps, and density maps for both dies of the optimized and unoptimized designs using the LDPC benchmark. In the optimized design, cells are more effectively distributed, leading to improved congestion and density without significantly increasing wirelength (Table III), due to reduced detours. This translates into substantial full-chip PPA improvements at the end of the flow, as detailed in Table III.

To address limited congestion awareness in the standard Pin-3D flow, we introduce two enhanced benchmarks for comprehensive comparisons. The first enhancement, "Pin-3D + Cong.", enables *ICC2* congestion-driven placement at the highest effort during the 3D placement stage. The second benchmark, "Pin-3D + BO," augments Pin-3D with Bayesian Optimization (BO) techniques from [19] to optimize tool parameters listed in Table I.

The detailed optimization results of Pin-3D, its variants, and our DCO-3D are presented in Table III. While DCO-3D demonstrates immediate congestion optimization benefits following the 3D placement stage, the key achievement lies in translating these benefits into superior 3D PPA outcomes at the end of the 3D signoff step. Across six industrial benchmarks using a foundry $3nm$ technology node, DCO-3D consistently delivers the best end-of-flow PPA metrics, with setup TNS improved by up to 86% and total power reduced by up to 4.8%. These results highlight how our early congestion optimization leads to significant full-chip PPA improvements in 3D ICs.

C. Why Does DCO-3D Work While Others Do Not?

Unlike "Pin-3D + Cong." and "Pin-3D + BO," which rely on ad-hoc congestion optimization, DCO-3D introduces differentiable congestion objectives optimized directly via gradient-based methods. By enabling cross-die cell spreading, DCO-3D redistributes congestion and substantially reduces routing overflow. Furthermore, by co-optimizing congestion with other objectives, DCO-3D minimizes congestion without compromising overall design quality. As shown in Table III, DCO-3D achieves notable PPA improvements, outperforming BO and commercial auto-fixing features, which often compromise critical PPA metrics.

VI. Conclusion

In this paper, we introduce DCO-3D, the first 3D flow that employs ML techniques to tackle routability issues in advanced technology nodes. DCO-3D accurately forecasts routability during the early stages of 3D global placement and a implements differentiable 3D cell spreading treatment. Extensive experiments show that DCO-3D not only effectively reduces congestion but also significantly improves PPA metrics during the final signoff evaluation.

VII. Acknowledgement

This work was supported by the Semiconductor Research Corporation under the JUMP 2.0 Center Program (CHIMES), Samsung Advanced Institute of Technology (SAIT) under the AI for Semiconductors Program, and the Ministry of Trade, Industry & Energy of South Korea (1415187652, RS-2023-00234159).

References

[1] H. Park, B. W. Ku, K. Chang, D. E. Shim, and S. K. Lim, “Pseudo-3d approaches for commercial-grade rtl-to-gds tool flow targeting monolithic 3d ics,” in *Proceedings of the 2020 International Symposium on Physical Design*, pp. 47–54, 202.

[2] W.-T. J. Chan, P.-H. Ho, A. B. Kahng, and P. Saxena, “Routability optimization for industrial designs at sub-14nm process nodes using machine learning,” in *Proceedings of the 2017 ACM on International Symposium on Physical Design*, pp. 15–21, 2017.

[3] W.-T. J. Chan, Y. Du, A. B. Kahng, S. Nath, and K. Samadi, “BEOL stack-aware routability prediction from placement using data mining techniques,” *2016 IEEE 34th International Conference on Computer Design (ICCD)*, pp. 41–48, 2016.

[4] Z. Xie, Y.-H. Huang, G.-Q. Fang, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu, “RouteNet: Routability prediction for Mixed-Size Designs Using Convolutional Neural Network,” in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2018.

[5] S. Kim, H. Park, K. Baek, K. Choi, and T. Kim, “Methodology of Resolving Design Rule Checking Violations Coupled with Fully Compatible Prediction Model,” in *Proceedings of the 2024 International Symposium on Physical Design*, ISPD ’24, (New York, NY, USA), p. 103–111, Association for Computing Machinery, 2024.

[6] R. Liang, H. Xiang, D. Pandey, L. Reddy, S. Ramji, G.-J. Nam, and J. Hu, “DRC Hotspot Prediction at Sub-10nm Process Nodes Using Customized Convolutional Network,” in *Proceedings of the 2020 International Symposium on Physical Design*, ISPD ’20, (New York, NY, USA), p. 135–142, Association for Computing Machinery, 2020.

[7] C. Yu and Z. Zhang, “Painting on Placement: Forecasting Routing Congestion using Conditional Generative Adversarial Nets,” in *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC ’19, (New York, NY, USA), Association for Computing Machinery, 2019.

[8] M. B. Alawieh, W. Li, Y. Lin, L. Singhal, M. A. Iyer, and D. Z. Pan, “High-Definition Routing Congestion Prediction for Large-Scale FPGAs,” in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 26–31, 2020.

[9] Y.-H. Huang, Z. Xie, G.-Q. Fang, T.-C. Yu, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu, “Routability-Driven Macro Placement with Embedded CNN-Based Prediction Model,” pp. 180–185, 03 2019.

[10] G. Koch, R. Zemel, R. Salakhutdinov, *et al.*, “Siamese neural networks for one-shot image recognition,” in *ICML deep learning workshop*, vol. 2, Lille, 2015.

[11] S. S. Kiran Pentapati *et al.*, “Pin-3D: A Physical Synthesis and Post-Layout Optimization Flow for Heterogeneous Monolithic 3D ICs,” in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2020.

[12] J. Cong and G. Luo, “A multilevel analytical placement for 3D ICs,” in *2009 Asia and South Pacific Design Automation Conference*, pp. 361–366, 2009.

[13] 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)*, pp. 1–2, 2023.

[14] J. Lu, H. Zhuang, I. Kang, P. Chen, and C.-K. Cheng, “ePlace-3D: Electrostatics based Placement for 3D-ICs,” in *Proceedings of the 2016 on International Symposium on Physical Design*, ISPD ’16, (New York, NY, USA), p. 11–18, Association for Computing Machinery, 2016.

[15] M.-K. Hsu, V. Balabanov, and Y.-W. Chang, “TSV-Aware Analytical Placement for 3-D IC Designs Based on a Novel Weighted-Average Wirelength Model,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 4, pp. 497–509, 2013.

[16] X. Zhao, S. Chen, Y. Qiu, J. Li, Z. Huang, B. Xie, X. Li, and Y. Bao, “iPL-3D: A Novel Bilevel Programming Model for Die-to-Die Placement,” in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–9, 2023.

[17] H. Park *et al.*, “Pseudo-3D Physical Design Flow for Monolithic 3D ICs: Comparisons and Enhancements,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 26, jun 2021.

[18] P. Spindler and F. M. Johannes, “Fast and Accurate Routing Demand Estimation for Efficient Routability-driven Placement,” in *2007 Design, Automation & Test in Europe Conference & Exhibition*, pp. 1–6, 2007.

[19] Y. Ma, Z. Yu, and B. Yu, “CAD Tool Design Space Exploration via Bayesian Optimization,” in *2019 ACM/IEEE 1st Workshop on Machine Learning for CAD (MLCAD)*, pp. 1–6, 2019.