

Netlist Tomography-Based Soft Guide Generation with Refinement Across P&R Stages for Improved PPA

Andrew B. Kahng
abk@ucsd.edu
UC San Diego
La Jolla, CA, USA

Sayak Kundu
sakundu@ucsd.edu
UC San Diego
La Jolla, CA, USA

Yiting Liu
yil375@ucsd.edu
UC San Diego
La Jolla, CA, USA

Dooseok Yoon
d3yoon@ucsd.edu
UC San Diego
La Jolla, CA, USA

Abstract

Timing closure remains the most critical challenge in physical design, as place-and-route (P&R) tools often lack guidance on which instances should be co-located to minimize path delays. We introduce a novel *Netlist Tomography* framework to improve power, performance and area outcomes in challenging P&R contexts. Our approach runs multiple placements of the same synthesized netlist in parallel, each on a slightly perturbed floorplan, to reveal groups of cells that consistently remain together and paths that are consistently timing-critical. These runs guide *TomoGNN*, a self-supervised graph model to learn instance embeddings that encode stable spatial-timing relationships. HDBSCAN-based clustering then converts the learned instance embeddings into *soft guides* – placement hints that encourage beneficial co-location without restricting the tool. These soft guides are adaptively refined across later P&R stages as the netlist and its critical paths evolve. Experimental results in NanGate 45 nm and GlobalFoundries 12 nm demonstrate up to (geo-mean) 42% (12%), 37% (18%) and 2.0% (0.3%) improvements in worst negative slack, total negative slack and total power when used to augment a leading P&R tool flow.

CCS Concepts

• **Hardware** → **Physical design (EDA)**; • **Computing methodologies** → *Machine learning*.

Keywords

physical design, placement, soft guides, clustering, graph neural networks, timing closure

ACM Reference Format:

Andrew B. Kahng, Sayak Kundu, Yiting Liu, and Dooseok Yoon. 2026. Netlist Tomography-Based Soft Guide Generation with Refinement Across P&R Stages for Improved PPA. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '26)*, November 08–12, 2026, San Jose, CA, USA. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3831252.3834239>

1 Introduction

Meeting timing requirements remains one of the most critical and time-consuming challenges in physical design (PD), as chips must satisfy performance targets under diverse process, voltage, and temperature (PVT) conditions. Achieving timing closure demands precise placement of timing-critical instances, yet place-and-route (P&R) tools often lack effective guidance on which instances should be co-located to minimize path delay. Because timing-critical paths can span the entire die, suboptimal placement leads to violations



Fig. 1: Netlist tomography views of the CA53 design on GF12. Each view is a placeOpt result on a slightly perturbed floorplan. Instances of the same logical hierarchy (same color) remain consistently co-located across views, motivating the use of cross-view stability to generate soft guides.

that require iterative and costly fixes. As technology nodes continue to shrink, these issues intensify, emphasizing the need to condition the placement problem for a given netlist by generating effective placement guidance – e.g., soft co-location constraints – to help achieve reliable and efficient timing closure.

Use of soft guide lever in P&R. Soft guides are soft placement constraints in P&R tools that cluster selected cells without fixing their exact locations. In Cadence Innovus v21.1, they are created using the *createInstGroup* and *createSoftGuide* commands or the DEF *GROUP* syntax. By enabling effective cell clustering, soft guides improve placement convergence and can yield better timing, lower power, and more efficient routing. In our methodology, soft guides are the primary lever for conditioning the placement problem. (Our background studies found other levers, such as cell padding and placement blockages, far less effective.) Identifying soft guides that improve power, performance, and area (PPA) is the central challenge addressed in this work. We focus on total negative slack (TNS) as the primary timing metric, as it is generally more indicative of remaining timing-closure effort than worst negative slack (WNS). **Concept of netlist tomography.** Identifying cell clusters that improve PPA is challenging, as it is difficult to predict which instances should remain together for optimal results. Here we propose *netlist tomography*, which creates multiple netlist views by perturbing floorplan parameters such as height, width, and row orientation. For each perturbed floorplan, we run placeOpt and record cell locations and slack values. Analyzing these views reveals instances that consistently remain co-located, or timing arcs that are consistently critical; our hypothesis is that this can lead us to effective soft guide candidates. Since placeOpt is less expensive relative to clock tree synthesis (CTS), routing, and optimization steps, *producing these additional views with parallel placeOpt runs adds modest wall time while yielding valuable information*. Figure 1 shows varying logical hierarchy placements across five perturbed floorplans (i.e., *netlist tomography views*) for the Arm Cortex-A53 (CA53) design.

In the PD flow, the netlist evolves at every stage, as optimizations introduce logic restructuring and new buffers or inverters. This new logic does not inherit existing soft guide constraints, potentially disrupting downstream placement. Figure 2 shows how the top 500 critical-path endpoints vary across the post-placeOpt, clockOpt, and routeOpt stages for the Ariane, CA53, and BlackParrot quad-core (BP) designs. Most endpoints are unique to each stage, with



This work is licensed under a Creative Commons Attribution 4.0 International License. ICCAD '26, San Jose, CA, USA

© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2873-0/2026/11
<https://doi.org/10.1145/3831252.3834239>

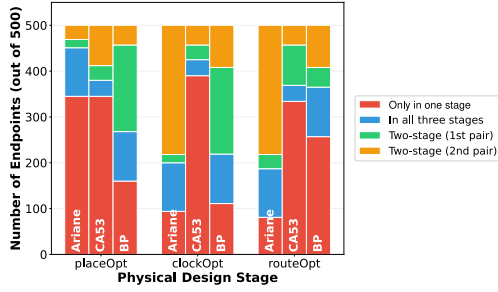


Fig. 2: Top 500 critical-path endpoint distribution for the Ariane, CA53 and BP designs across the (successive) post-placeOpt, clockOpt and routeOpt stages of the P&R flow. Most endpoints are unique to each stage, with limited overlap (blue), showing that critical paths shift as the netlist evolves and motivating refinement of soft guides at each stage.

limited overlap (in blue), indicating significant critical-path changes across the flow. Because soft guides generated based on the placeOpt views no longer align with the shifting critical paths at clockOpt and routeOpt, updating them becomes essential to maintain relevance, guide downstream optimization, and improve PPA. Against this backdrop, the key contributions of our work are summarized as follows.

- We introduce *netlist tomography*, a placeOpt-driven approach that generates multiple views of a synthesized netlist. These views are used in an ML pipeline to create soft guides that improve routability and post-route performance.
- We construct *co-location graphs* from netlist tomography views to capture consistent physical proximity across placement scenarios. Building on this graph, we develop *TomoGNN*, a tomography-aware graph model trained with PPA-driven loss that accounts for timing slack, physical location, and logical hierarchy to learn embeddings for the synthesized netlist. These embeddings are then processed with HDBSCAN to generate high-quality soft guides for P&R flow.
- We introduce a soft guide refinement flow that keeps guides aligned with target paths at each optimization stage of the P&R flow (i.e., placeOpt, clockOpt), thereby improving post-route performance.
- We evaluate our approach on the open-source NanGate45 (NG45) library [35] and the commercial GlobalFoundries 12 nm (GF12) enablement, using the open designs JPEG, SweRV, Ariane and Black-Parrot (Quad-Core), and the commercial design Arm Cortex-A53 (CA53), with the Cadence Innovus v21.1 tool as the evaluator.¹ We have made the code and scripts used in this work available on GitHub [36], along with additional studies and supporting results, enabling full reproduction of the proposed flow and the TomoGNN implementation.²

2 Related Work

Clustering is widely used in physical design to reduce problem size and improve PPA and runtime. Multilevel methods such as FastPlace [30] use two-level clustering for coarsening. Other approaches include best-choice [2] [3] and fine-granularity [14] clustering, hierarchy-guided partitioning [10], and modularity-based [12, 29]

Table 1: Notation

Term	Description
$G = (V, E)$	Netlist graph (nodes V , edges E)
N_i	Set of neighbors of node i
X	Node physical locations
ℓ_{ij}	Physical length of edge (i, j)
s_{ij}, c_{ij}	Slack and criticality for edge (i, j)
T_{clk}	Reference clock period
$\mathcal{P}$	Set of timing paths
$L_{ij}, P_{95}, L_{\text{max}}$	Normalized length, 95th percentile, clipping threshold
h_{ij}	Hierarchy weight of edge (i, j)
ω_{ij}^+	Attraction (+) and repulsion (-) weights for edge (i, j)
p	Criticality exponent for attraction strength
β	Decay parameter for length-based repulsion
sim_{ij}	Embedding similarity for nodes i and j
$\mathcal{P}^+, \mathcal{N}^-$	Positive and negative pairs for contrastive learning
$\mathcal{L}, \lambda_{\text{push}}, \lambda_{\text{cal}}$	Total loss function and balance parameters
$X, x_i^{(m)}$	All node features and hidden representation of node i at layer m
$Z_\theta, z^*, \hat{z}_i$	Node embeddings (matrix, per-run, fused)
H^y, y, R	Co-location graph per run, run index, total runs
r_{adapt}	Adaptive radius for co-location graph
w_{ij}	Edge weight of the co-location graph
π_y	Run quality score for weighting
$M, m, \alpha^{(m)}$	Number of GNN layers, layer index and gating parameter
W, q	Learnable weight matrix, learnable attention parameter
N	Number of nearest neighbors for refinement
d_0, d', d	Initial, hidden and final feature dimensions
$\parallel$	Vector concatenation
a_i^y	Attention weight for node i in run y

and density-based [24] clustering. Complementary techniques include graph centrality analysis [33], PPA-aware clustering [17], SafeChoice [32], hypergraph partitioning [5], hierarchical RTL-based methods [19] [8] [9], and GPU-accelerated optimization [21]. Recent works on graph and hypergraph partitioning, such as ParS-GCN [31], and RL-based initialization methods [26], provide strong advances in balanced partitioning with application-specific objectives (e.g., cut minimization or scheduling efficiency). However, these partitioning approaches typically enforce assignment of all instances into a fixed number of roughly balanced partitions, which differs from the soft guide setting where (i) cluster sizes are inherently variable and driven by PPA relevance, and (ii) outlier instances should remain unclustered when no reliable grouping exists. Moreover, these methods do not adapt to evolving topological or timing considerations as optimization progresses, and their objectives do not directly capture physical design metrics such as timing slack, spatial proximity, or hierarchy.

Graph neural networks offer a data-driven alternative by learning netlist representations that can inform clustering. Foundational architectures such as GraphSAGE [13] enable scalable inductive learning on large graphs, and hypergraph convolution [4] extends message passing to multi-pin nets. Building on these, several GNN-based methods have been proposed for placement-oriented clustering, including Law of Attraction [22], PPA-directed clustering [23] [15], ClusterNet [25], noise-aware clustering [7], PTA-GNN [11], and GOALPlace [1]. While GOALPlace incorporates post-routeOpt features, it retains only instances matching synthesis-netlist names, limiting its ability to handle netlist evolution. These methods generate static clusters from single design snapshots without dynamic adaptation.

Placement tomography [18] conditions the routing problem for a given (i.e., fixed) placement by predicting routing hotspots from multiple placement views. In contrast, netlist tomography conditions the placement problem for a given (i.e., fixed) synthesized netlist. Three elements distinguish our method from prior work: (1) netlist tomography extracting timing, proximity and hierarchy features across placement views; (2) PPA-driven contrastive loss that combines geometric and hierarchical constraints beyond single-objective methods [22, 23, 25]; and (3) three-phase refinement that

¹We do not perform any benchmarking of commercial EDA tools.

²Due to IP restrictions, the repo excludes GF12 enablement files, as well as any design files or information associated with the CA53 design.

adaptively updates clusters throughout P&R to handle netlist evolution, unlike static methods [7, 12, 23, 26, 31]. To the best of our knowledge, ours is the first approach that combines netlist tomography with adaptive, timing-aware soft guide refinement across P&R stages to improve performance.

3 Problem Formulation

Given a synthesized netlist $G = (V, E)$, a target floorplan F , and R tomography runs, each run γ providing features $X^{(\gamma)}$ capturing timing, hierarchy and location, we seek cluster assignments C from learned embeddings that aggregate information across all R runs. The target floorplan F is the baseline floorplan obtained after synthesis and macro placement, before soft guide generation. The same floorplan F is used by the default P&R flow and by the final soft-guide-guided P&R flow.

These learned embeddings minimize a PPA-driven loss function $\mathcal{L}_{\text{PPA}}$ that accounts for timing criticality, physical proximity, and hierarchical relationships. Multi-run features are aggregated with quality-based weighting to capture consistent co-location patterns. The clusters concentrate timing-critical instances while respecting floorplan constraints, generating soft guides that improve PPA throughout the physical design flow.

We train a tomography-aware graph model f to produce embeddings that minimize the PPA-driven loss function:

$$\begin{aligned} \theta^* &= \arg \min_{\theta} \mathcal{L}_{\text{PPA}}(Z_{\theta}, t), \\ Z_{\theta^*} &= f_{\theta^*}(G, X, F), \\ C &= \text{Cluster}(Z_{\theta^*}). \end{aligned} \quad (1)$$

The learned embeddings map instances into a space where those with similar PPA impact cluster together, serving as soft guides for the physical design tools.

4 Our Approach

Figure 3 illustrates our approach. For a given synthesized netlist and target floorplan, we generate netlist tomography features by perturbing the floorplan and running *place_opt_design* for each perturbation in parallel. We train TomoGNN using a PPA-driven contrastive loss to learn node embeddings, then apply HDBSCAN clustering to generate soft guides. During P&R, we refine the soft guides after placeOpt and clockOpt to maintain alignment with evolving critical paths. The following subsections describe each step in detail.

4.1 Netlist Tomography Feature Extraction

To generate netlist tomography features, we perturb the target floorplan, run placement optimization, and extract placement-based features from each perturbation. We create 30 perturbed floorplans by varying the height by ± 1 row heights, the width by ± 2 site widths, and the first row orientation (R0 or MX). Each perturbation is applied to the global core dimensions of the target floorplan. We choose these values because they are the smallest legal placement-grid perturbations that create meaningful placement diversity, based on our background studies of P&R tool behaviors. The perturbed floorplans are used only for tomography feature extraction; the final soft-guide-guided P&R flow runs on the original target floorplan. For each perturbed floorplan, we run *place_opt_design* to obtain a placement solution. From each placement, we extract the netlist graph (i.e., source-to-sink graph), instance locations, edge

slack values, and reference clock period. The reference clock period is captured together with the slack to ensure accurate timing-criticality analysis in designs that contain multiple clocks (e.g., BlackParrot). These multi-view features collectively form our netlist tomography feature set. To better capture the physical proximity information across multiple tomography runs, we construct a *co-location graph* H . It connects spatially proximate instances within an adaptive radius $r_{\text{adapt}} = q \cdot \text{median}(1\text{-NN distances})$ with edge weights w_{ij} determined by physical distance $\ell_{i,j}$. Specifically, we randomly sample 10K instances and compute the Manhattan distance to each sampled instance's nearest neighbor (i.e., 1-NN distances). We then take the median of these 1-NN distances and scale it by a factor q ($q = 3.0$) to obtain the adaptive radius. The sampled instances are used only to estimate r_{adapt} , since computing 1-NN distances over all instances is expensive for large designs. Sweeping the sample size $M_s \in \{1K, 5K, 10K, 20K, 30K\}$ across all 10 testcases changes the estimated radius by at most 0.61%; we therefore use $M_s = 10K$ as a conservative default. The edge weight $w_{ij} = \exp(-(\ell_{ij}/\sigma)^2)$ encodes spatial closeness between instances, where σ is a distance scaling factor that controls how fast the weight decays with distances. Based on background studies, we empirically set $\sigma = 10$ in this work.

Mapping features to the synthesis netlist. During placement optimization, the netlist undergoes transformations such as buffer insertion or deletion, gate sizing, and logic restructuring, which cause mismatches between the placeOpt and synthesis netlists. To address this mismatch, Algorithm 1 maps features from the placeOpt netlist A to the synthesis netlist B . Lines 1–2 identify matched nodes using instance names. For unmatched nodes (Lines 3–6), we perform bidirectional multi-source BFS to locate connected nodes with known positions. Lines 7–9 estimate the location of each unmatched node as the average position of its transitive fanins and fanouts (found via BFS backwards and forward), while assigning the minimum slack from its transitive fanins. Edge slack values are mapped in a similar manner, and unmapped edges are assigned the minimum slack among their connected nodes.³

Algorithm 1: Mapping placeOpt (A) to synthesis (B).

```

Input:  $N_A$ : nodes from placeOpt with known locations,
 $N_B$ : nodes from synthesis needing estimation,  $G = (V, E)$ : synthesis netlist graph
Output: Estimated locations for all nodes in  $B$ 
1  $N_{\text{match}} \leftarrow N_A \cap N_B$  // Matched nodes
2  $M \leftarrow N_B \setminus N_{\text{match}}$  // Unmatched nodes
3 foreach  $v$  in  $M$  do
4    $F_{\text{in}}(v) \leftarrow \text{FindFanins}(v, G, N_{\text{match}})$  // BFS backward
5    $F_{\text{out}}(v) \leftarrow \text{FindFanouts}(v, G, N_{\text{match}})$  // BFS forward
6    $N(v) \leftarrow F_{\text{in}}(v) \cup F_{\text{out}}(v)$  // Connected nodes
7   if  $N(v) \neq \emptyset$  then
8      $\hat{x}_v \leftarrow \frac{1}{|N(v)|} \sum_{u \in N(v)} x_u$ ;  $\hat{y}_v \leftarrow \frac{1}{|N(v)|} \sum_{u \in N(v)} y_u$ ;
9      $\text{slack}_v \leftarrow \min_{w \in F_{\text{in}}(v)} \text{slack}_w$  // Worst slack

```

4.2 TomoGNN Model

Our **TomoGNN** is a tomography-aware graph neural network that learns stable spatial–timing relationships from multiple placement runs. TomoGNN processes two complementary graphs for each run γ : the *netlist graph* $G = (V, E)$ representing logical connectivity, and the *co-location graph* $H^\gamma = (V, E_H^\gamma)$ capturing physical proximity in each run. Figure 4 illustrates the complete TomoGNN pipeline. The

³In our GitHub [36], *netlist_tomo_utils.tcl* extracts features from the Innovus DB, and *batch_synth_mapping.py* maps the placeOpt features back to the synthesized netlist.

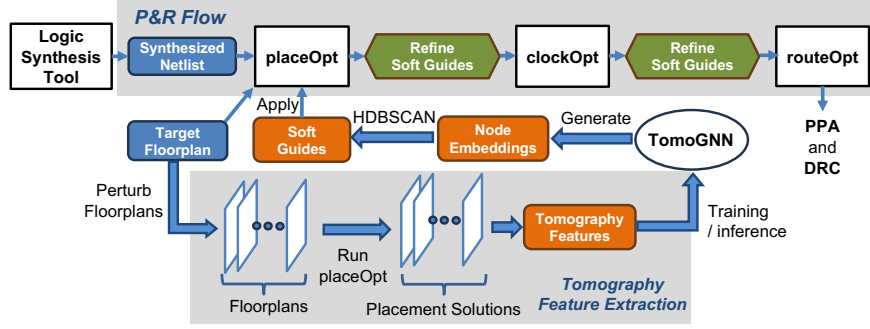


Fig. 3: Overview of our netlist tomography-based soft guide generation and refinement approach. The Target Floorplan is the baseline floorplan obtained after synthesis and macro placement.

architecture processes netlist and co-location graphs through dual-graph message passing⁴ with run-conditioned feature modulation, producing per-run node embeddings. These per-run embeddings are then fused via a quality-aware attention mechanism to obtain the single embedding for each node. Note that the netlist graph G is shared across all runs (as it represents the same circuit), while each co-location graph H^γ is run-specific, capturing physical proximity from that run's placement.

Architecture. TomoGNN treats each placement run as a separate view and employs three key mechanisms: (1) run-conditioned feature modulation via FiLM (Feature-wise Linear Modulation) [28], which generates modulated per-run node features x_i^γ ; (2) dual-graph message passing, which aggregates information from both graphs; and (3) quality-aware view fusion, which assigns attention weights to run γ by its timing quality score $\pi_\gamma = 1.0/(1.0 + |\text{TNS}_\gamma|/\tau_q)$, where TNS_γ is the TNS of run γ . We set the normalization constant τ_q to 3000 based on the empirical TNS distribution observed across our placement runs.

Run-conditioned feature modulation. Each tomography run corresponds to a different floorplan perturbation, resulting in variations in placement and timing behavior. To explicitly capture these run-specific effects, we apply FiLM to condition node features on the run context. Let $\mathbf{x}_i \in \mathbb{R}^d$ denote the base feature vector of node i . FiLM injects run identity into the feature space through a run-dependent affine transformation:

$$\mathbf{x}_i^\gamma = \boldsymbol{\alpha}^{(\gamma)} \odot \mathbf{x}_i + \boldsymbol{\beta}^{(\gamma)} \quad (2)$$

where $(\boldsymbol{\alpha}^{(\gamma)}, \boldsymbol{\beta}^{(\gamma)})$ are feature-wise scale and shift parameters generated via a lightweight MLP. The resulting $\mathbf{x}_i^\gamma$ is then used as the input to the dual-graph message passing layers for run γ .

Dual-graph message passing. For layer m , we use GraphSAGE with mean aggregation to compute messages from the netlist graph G and co-location graph H^γ :

$$\mathbf{z}_{i,G}^{(m,\gamma)} = W_G^{(m)} \left[\mathbf{x}_i^{(m,\gamma)} \parallel \frac{1}{|\mathcal{N}_G(i)|} \sum_{j \in \mathcal{N}_G(i)} \mathbf{x}_j^{(m,\gamma)} \right], \quad (3)$$

$$\mathbf{z}_{i,H}^{(m,\gamma)} = W_H^{(m)} \left[\mathbf{x}_i^{(m,\gamma)} \parallel \frac{1}{|\mathcal{N}_{H^\gamma}(i)|} \sum_{j \in \mathcal{N}_{H^\gamma}(i)} w_{ij}^{(\gamma)} \mathbf{x}_j^{(m,\gamma)} \right], \quad (4)$$

where the co-location edge weights $w_{ij}^{(\gamma)}$ modulate neighbor contributions. The two messages are fused using a learnable gate $\alpha^{(m)}$,

⁴Information is propagated over two different but related graphs, instead of just one.

followed by layer normalization $\text{LN}(\cdot)$:

$$\mathbf{x}_i^{(m+1,\gamma)} = \text{LN} \left(\mathbf{x}_i^{(m,\gamma)} + \text{LN}(\mathbf{z}_{i,G}^{(m,\gamma)}) + \alpha^{(m)} \text{LN}(\mathbf{z}_{i,H}^{(m,\gamma)}) \right). \quad (5)$$

After $M = 3$ layers, the per-run embedding is obtained via L2 normalization $\mathbf{z}_i^{(\gamma)} = \mathbf{x}_i^{(M,\gamma)} / \|\mathbf{x}_i^{(M,\gamma)}\|_2$.

Quality-aware view fusion. Given per-run embeddings $\mathbf{z}_i^{(\gamma)}$, TomoGNN computes an attention weight $a_i^{(\gamma)}$ for each run:

$$a_i^{(\gamma)} = \text{softmax}_\gamma \left(q^\top \tanh(W \mathbf{z}_i^{(\gamma)}) + \log \pi_\gamma \right), \quad (6)$$

where q is a learnable attention parameter, W is a learnable weight matrix, and $\log \pi_\gamma$ is the quality-based bias so that higher-quality runs receive larger attention.

The final fused embedding for each node i is:

$$\hat{\mathbf{z}}_i = \sum_{\gamma=1}^R a_i^{(\gamma)} \mathbf{z}_i^{(\gamma)}. \quad (7)$$

By integrating information from both logical connectivity and multi-run physical proximity, TomoGNN learns embeddings that capture stable, performance-critical spatial–timing relationships, rather than noise from any single placement run.

Embedding generation. After training, we generate final embeddings for all nodes using full-neighborhood inference (fanout = -1). We process nodes in mini-batches and stitch embeddings by mapping to global node IDs. The resulting embeddings $\hat{\mathbf{z}}$ are L2-normalized vectors, which are further processed for downstream soft guide generation.

4.3 PPA-Driven Loss and Training

We design a PPA-driven loss function combining contrastive learning with geometric constraints to learn embeddings that capture timing criticality and physical proximity. For each edge (i, j) , we compute criticality $c_{ij} = \min(1, \max(0, -s_{ij})/T_{\text{clk}})$, length $L_{ij} = \min(L_{\text{max}}, \ell_{ij}/P_{95})$, and hierarchy weight $h_{ij} = (2 \cdot d_{\text{LCA}}/\max(d_i, d_j)) - 1$, where s_{ij} is the edge slack, T_{clk} is the clock period, ℓ_{ij} is the Manhattan distance, d_i is the hierarchy depth of the instance i and d_{LCA} is the lowest common ancestor (LCA) depth in the module hierarchy.⁵ We define attraction and repulsion strengths as $f_{ij} = c_{ij}^p$ and $g_{ij} = e^{-\beta L_{ij}}$, where p is the criticality exponent and β is the length-decay parameter, yielding edge weights:

$$\omega_{ij}^+ = f_{ij} \cdot g_{ij} + h_{ij}, \quad \omega_{ij}^- = (1 - f_{ij}) \cdot (1 - g_{ij}) - h_{ij}. \quad (8)$$

⁵In our experiments, $d_i, d_j \in [1, 7]$ and are typically similar ($|d_i - d_j|$ has a median of 0 and a 95th percentile of 1), while $d_{\text{LCA}} \in [0, 7]$; the resulting $h_{ij} \in [-1, 0.72]$.

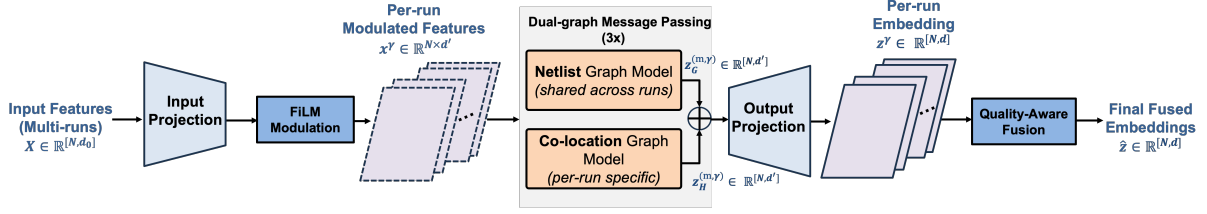


Fig. 4: TomoGNN architecture.

Thus, ω_{ij}^+ is larger for short, highly critical, same-hierarchy edges, whereas ω_{ij}^- emphasizes their complementary counterparts. Given learned node embeddings $\hat{z}_i$, we sample positive pairs $\mathcal{P}^+$ and negative pairs $\mathcal{N}^-$ based on ω^+ and ω^- . The weighted InfoNCE loss [27] with similarity $\text{sim}_{ij} = \hat{z}_i^\top \hat{z}_j / \tau$, where τ is a learnable temperature parameter, and partition function $\mathcal{Z} = \sum_{(m,n) \in \mathcal{P}^+} e^{\text{sim}_{mn}} + \sum_{(m,n) \in \mathcal{N}^-} e^{\text{sim}_{mn}}$ is:

$$\mathcal{L}_{\text{InfoNCE}} = - \sum_{(i,j) \in \mathcal{P}^+} \omega_{ij}^+ \log \frac{e^{\text{sim}_{ij}}}{\mathcal{Z}}. \quad (9)$$

To push negative pairs below threshold $-\mu$, we add a smooth margin-based repulsion term:

$$\mathcal{L}_{\text{push}} = \sum_{(i,j) \in \mathcal{N}^-} \omega_{ij}^- \cdot \text{softplus} \left(\frac{\text{sim}_{ij} + \mu}{\tau_p} \right), \quad (10)$$

where $\mu = 0.10$ and $\tau_p = 0.5$ control the margin sharpness. Together, $\mathcal{L}_{\text{InfoNCE}}$ and $\mathcal{L}_{\text{push}}$ guide the model to pull PPA-relevant node pairs closer while pushing less relevant ones apart, producing a discriminative and clusterable embedding space for downstream soft guide generation.

Note that the contrastive loss shapes the relative geometry of the embedding space; it does *not* explicitly ensure consistency with physical proximity observed across multiple runs. To this end, we apply a tomography calibration loss:

$$\mathcal{L}_{\text{cal}} = \sum_{(i,j)} \text{BCE}(P_{ij}, C_{ij}), \quad (11)$$

where BCE is Binary Cross-Entropy, $P_{ij} = \text{sigmoid}(\text{sim}_{ij} / \tau_c)$ is the predicted co-location probability with $\tau_c = 0.6$, and C_{ij} is the target co-location probability, computed as a quality-weighted average across runs, i.e.,

$$C_{ij} = \frac{\sum_Y \pi_Y \exp(-(\ell_{ij}^{(Y)} / \sigma)^2)}{\sum_Y \pi_Y} \quad (12)$$

with $\ell_{ij}^{(Y)}$ being the Manhattan distance in run Y . The score $C_{ij} \in [0, 1]$ indicates how frequently nodes are co-located: $C_{ij} \rightarrow 1$ for nodes often close, $C_{ij} \rightarrow 0$ for those rarely close.

The total loss is $\mathcal{L} = \mathcal{L}_{\text{InfoNCE}} + \lambda_{\text{push}} \mathcal{L}_{\text{push}} + \lambda_{\text{cal}} \mathcal{L}_{\text{cal}}$, where the weights λ_{push} and λ_{cal} balance the loss terms and are scheduled during training as described below.

Training. We train TomoGNN⁶ using neighbor sampling with fanouts {15, 10, 5} and batch size 4096 nodes. We optimize using AdamW [20] with learning rates 8×10^{-3} (model) and 1×10^{-4} (temperature), weight decay 10^{-4} , and ReduceLROnPlateau scheduling (factor 0.7, patience 20). The model is trained for a maximum of 1000 epochs, with early stopping applied to prevent overfitting if validation performance does not improve for 50 consecutive epochs.

⁶In our GitHub [36], *train.py* provides the implementation of the model training.

Loss weights are dynamically adjusted: λ_{push} scales with the temperature after epoch 50, and λ_{cal} linearly increases from 0.3 to 1.0 over epochs 50-150 and remains at 1.0 thereafter. For efficiency, we randomly sample 3 of the 30 runs per epoch.

4.4 Soft Guide Generation and Refinement

Soft guide generation. We apply HDBSCAN [24] to cluster the learned embeddings, using minimum cluster size of 200 and minimum sample size of 5. HDBSCAN discovers clusters of varying densities and automatically identifies outliers, well-suited for heterogeneous netlist graphs. The resulting clusters define soft guides that specify which instances should be placed near each other without prescribing exact locations, maintaining placement tool flexibility while encouraging co-location of timing-critical instances.

Soft guide refinement. During P&R, logic restructuring creates unmapped instances and changes criticality, requiring iterative refinement. We apply Algorithm 2 after *place_opt_design* and *ccopt_design* to reassign unmapped instances, refine boundary instances, and merge fragmented clusters. **Lines 1–5** assign unmapped instances to cluster based on a scoring function $(n + 1) / (\text{dist} + 1)$ that balances connectivity n and Euclidean distance dist from the instance to the cluster centroid. **Lines 6–17** refine boundary instances using the gain expression $(r - a) - \delta \times \kappa$, where r and a represent cuts removed and added, δ is the distance penalty, and κ denotes criticality; this process iterates until convergence. **Lines 18–26** merge clusters that have ≥ 400 path transitions and spatial proximity $\text{dist} < 1.1(r_{g_1} + r_{g_2})$, where dist is the Euclidean distance between the two cluster centroids and r_g is the average Euclidean distance from instances in cluster g to its centroid, and that satisfy the size constraint $|g_1| + |g_2| \leq 1.1 \times \text{max_size}$, where max_size is the maximum size among the initial clusters. This three-phase refinement, performed after *placeOpt* and *clockOpt*, helps ensure that guides remain aligned with the evolving netlist and criticality throughout P&R.⁷ The refinement relies on stage-specific observed information – the current placement, critical paths, connectivity, and cluster centroids and sizes – rather than PDK-specific tuning, and thus adapts to the design and flow stage at hand.

5 Experimental Setup and Results

We now describe our experiments that study the benefits of the approach. Ablation studies examine the benefits of tomography runs, the effectiveness of TomoGNN across varying target clock periods (TCPs), the design of the TomoGNN architecture, our soft guide refinement method and the benefit of soft guides over additional placement optimization. We also provide a detailed runtime breakdown.

⁷In our GitHub [36], *optimize_cluster.py* performs the refinement of soft guides.

Algorithm 2: Soft Guide Refinement

Input: Initial clusters C , node locations X , netlist graph G , timing paths $\mathcal{P}$
Output: Refined cluster assignments C'

```

1 foreach instance  $i$  with  $C[i] = -1$  do
2   Find at most  $N$  nearest clusters based on  $N$  nearest instances with cluster
   assignment;
3   foreach candidate cluster  $g$  do
4      $score(g) \leftarrow \frac{|neighbors(i) \cap g| + 1}{dist(i, centroid(g)) + 1}$ ;
5   Assign  $i$  to highest scoring  $g$  if within distance threshold and  $|g| < max\_size$ ;
6 for  $iter \leftarrow 1$  to  $max\_iterations$  do
7    $\mathcal{B} \leftarrow \{instances\ with\ neighbors\ in\ different\ clusters\}$ ;
8    $moved \leftarrow 0$ ;
9   foreach instance  $i \in \mathcal{B}$  do
10    Find nearby candidate clusters;
11    foreach candidate cluster  $g$  do
12       $gain(g) \leftarrow (cuts\_removed - cuts\_added)$ ;
13    if  $max(gain) > 0$  then
14      Move  $i$  to cluster with highest gain;
15       $moved \leftarrow moved + 1$ ;
16 if  $moved = 0$  then
17   break;
18 Compute cluster radius  $r_g$  for each cluster  $g$ ;
19 repeat
20   Count transitions between clusters on paths  $\mathcal{P}$ ;
21   foreach pair  $(g_1, g_2)$  with  $\geq 400$  transitions do
22     if  $dist(g_1, g_2) < 1.1(r_{g_1} + r_{g_2})$  and  $|g_1| + |g_2| \leq 1.1 \times max\_size$  then
23        $score(g_1, g_2) \leftarrow \frac{transitions(g_1, g_2)}{dist(g_1, g_2) + 1}$ ;
24   Merge pair with highest score;
25   Recompute radius for merged cluster;
26 until no valid candidates;
27 return  $C'$ ;

```

5.1 Experimental Setup

In this section, we present the benchmarks and PDKs used in our experiments, describe the evaluation flow, and outline the baselines used for comparison.

Benchmarks and PDKs. We conduct evaluations on the TILOS MacroPlacement [8] [9] [38] and OpenROAD-flow-scripts [37] designs (#inst, #macro) – JPEG (50k, 0), SweRV (100k, 28), Ariane (100k, 133), BlackParrot quad-core (BP) (768k, 220), and Arm Cortex-A53 (CA53) (350k, 20) – using both the NG45 [35] and GF12 PDKs.⁸ Netlists are synthesized with Cadence Genus v21.1, and macros are manually placed before performing P&R with Cadence Innovus v21.1. Across all designs, we tighten utilization to 70–80% to create challenging routing conditions. We also set the TCP to yield –50 to –150 ps post-routeOpt WNS in the baseline Innovus flow without soft guides. These tight utilization and TCP settings push the edge of commercial tool capability, making the baseline results very difficult to improve upon, and ensure that improvements from our method reflect meaningful enhancements rather than gains from relaxed default settings.

Evaluation flow. Figure 3 outlines our evaluation flow. We first perturb the target floorplan and run placeOpt to extract tomography features in parallel. We then conduct inference using the trained TomoGNN model on NVIDIA TITAN V (12 GB) GPUs and generate the soft guides. After inserting the soft guides, we run placeOpt followed by clockOpt and routeOpt. Following placeOpt and clockOpt, we extract the top 50K critical paths with unique start–end pairs (up to 20 paths per endpoint) and refine the soft guides as described in Section 4.4. P&R is executed with 16 threads on a server equipped with an Intel Xeon Gold 6148 CPU (2.40 GHz, 80 cores) and 384 GB of memory. We report post-routeOpt routed

⁸Details are provided only for the open-source designs on NG45. Our GitHub [36] contains all scripts needed to run Cadence Genus synthesis and Innovus P&R for these NG45 open-source designs.

wirelength (rWL), WNS, total negative slack (TNS), total power, the number of design rule violations (#DRV), and the runtime. To denoise P&R outcomes [6], every number in the tables below is derived from **five separate runs**, varying TCP by ± 0.2 ps in 0.1 ps steps and reporting the median value for each metric.⁹ This yields more stable results [16].

Baselines used for experimental evaluation. We evaluate the effectiveness of our TomoGNN soft guides with the following approaches:

- **Dflt:** The default Innovus flow without soft guides. The flow is timing- and congestion-driven. This serves as the primary reference for comparison and reflects current industrial practice without explicit clustering guidance.
- **SBase:** This baseline method reflects how engineers typically form soft guides to address timing-critical paths. Specifically, SBase creates soft guides from post-routeOpt timing paths in the default flow and clusters instances that frequently co-occur on these paths. The resulting clusters are mapped back to the synthesis netlist using Algorithm 1. Our code in [36] shows how SBase produces its guides.
- **Blob:** The state-of-the-art PPA-aware clustering method [17, 34]. Blob is the only recent work that open-sources its clustering code, which is why we use it as one of our baselines. For the Blob flow, we generate clusters from the placeOpt netlist and then apply Algorithm 1 to map them to the synthesized netlist. Although the default Blob flow [17] uses the synthesis netlist, we find that the placeOpt netlist gives better results. We hence use the placeOpt netlist in the Blob baseline.

Soft guides generated from both our method and the two baselines (SBase and Blob) pass through the same evaluation flow (i.e., placeOpt, refinement, clockOpt, refinement, and routeOpt). In the tables below, runtime of Dflt includes overall P&R runtime; SBase additionally includes one full default P&R run to obtain the post-routeOpt results, plus soft guide generation; Blob includes soft guide generation and the downstream P&R flow; and TomoGNN includes data extraction (i.e., tomography runs), model inference, soft guide generation and refinement, and the P&R stages (see Section 5.4).

5.2 Performance of Our TomoGNN Soft Guides

Table 2 presents the post-routeOpt routed wirelength, total power, WNS, TNS, #DRV, and runtime (RT) for the default tool flow (Dflt), two baselines (SBase and Blob) and our TomoGNN (T-GNN). For GF12, rWL and total power are normalized with respect to the default tool flow, and WNS and TNS are normalized to the target clock period. We make the following observations:

- Compared to the default tool flow, TomoGNN yields up to (geo-mean) 42% (12%) and 37% (18%) improvement in WNS and TNS, respectively. In seven of 10 cases for WNS and all 10 cases for TNS, TomoGNN achieves better results than the SBase flow, and in eight of 10 cases for both WNS and TNS, it surpasses the Blob flow, demonstrating that our PPA-driven, tomography-aware TomoGNN enhances design performance.
- Relative to the default flow, TomoGNN achieves similar power and routed wirelength, with geo-mean improvements of 0.3% and 0.1%, respectively, across the reported designs.
- TomoGNN is not designed as a standalone DRV optimizer; it provides soft timing-driven co-location guidance while leaving

⁹For the BP design, which has multiple clocks, we change the clock period of the main bp_clk by 0.1 ps and adjust the clock periods of the remaining clocks by maintaining their ratios with bp_clk .

Table 2: Post-routeOpt routed wirelength (rWL) (mm), power (mW), WNS (ps), TNS (ns), number of design rule violations (#DRV), and runtime (RT) (h) for tool default (Dflt), baselines (SBase and Blob), and TomoGNN.

Design	Type	rWL	Power	WNS	TNS	#DRV	RT
JPEG NG45	Dflt	499	317	-149	-86.6	0	0.49
	Blob	496	316	-139	-82.1	0	0.78
	SBase	496	317	-160	-90.8	0	1.07
	T-GNN	498	317	-119	-70.8	0	0.63
SweRV NG45	Dflt	3047	400	-87	-12.9	1	0.88
	Blob	3047	400	-87	-12.9	1	1.04
	SBase	3050	400	-87	-18.8	1	2.22
	T-GNN	3181	398	-76	-9.9	0	1.49
Ariane NG45	Dflt	4648	832	-115	-76.8	2	1.01
	Blob	4642	831	-129	-102.0	2	4.01
	SBase	4648	830	-117	-84.9	2	2.37
	T-GNN	4647	833	-111	-63.7	2	1.42
BP NG45	Dflt	25929	4470	-148	-321.9	0	5.28
	Blob	25993	4468	-107	-285.3	0	6.71
	SBase	25961	4470	-129	-294.0	0	11.39
	T-GNN	25960	4467	-138	-288.3	0	6.96
CA53 NG45	Dflt	11137	843	-61	-89.7	784	2.68
	Blob	11148	846	-49	-48.0	1391	5.81
	SBase	11109	844	-63	-77.4	1331	6.61
	T-GNN	11121	844	-53	-56.2	1057	3.97
JPEG GF12	Dflt	1.000	1.000	-0.565	-420	0	1.20
	Blob	0.940	0.992	-0.360	-419	0	1.64
	SBase	0.948	0.988	-0.350	-344	0	2.32
	T-GNN	0.948	0.980	-0.325	-335	0	1.46
SweRV GF12	Dflt	1.000	1.000	-0.244	-166	0	1.33
	Blob	1.006	1.000	-0.272	-181	0	3.05
	SBase	1.002	1.000	-0.242	-191	0	3.07
	T-GNN	1.004	0.993	-0.246	-158	0	2.80
Ariane GF12	Dflt	1.000	1.000	-0.123	-92	0	1.31
	Blob	1.002	1.006	-0.136	-133	0	3.79
	SBase	1.001	1.000	-0.128	-110	0	3.04
	T-GNN	0.999	1.000	-0.123	-92	0	1.71
BP GF12	Dflt	1.000	1.000	-0.135	-405	0	6.45
	Blob	0.999	0.998	-0.139	-489	1	9.99
	SBase	0.998	0.998	-0.134	-368	5	12.81
	T-GNN	1.002	1.000	-0.135	-265	0	7.22
CA53 GF12	Dflt	1.000	1.000	-0.213	-181	0	4.67
	Blob	1.001	1.001	-0.196	-172	1	7.96
	SBase	1.000	1.001	-0.232	-170	2	8.90
	T-GNN	1.000	0.999	-0.183	-162	1	4.88

detailed placement, routing, and DRV repair to the commercial P&R engine. Table 2 shows no systematic DRV degradation: TomoGNN has zero DRVs in seven of 10 design-node cases and improves SweRV (NG45) from one DRV to zero. For CA53 (NG45), our approach does not reduce #DRV relative to the default flow. It does, however, reduce #DRV by 24% (21%) compared with the Blob (SBase) flows.

- Across all designs, the default flow is faster by (geo-mean) 27%, as our method adds overhead from feature extraction, model inference, and soft guide refinement after the placeOpt and clockOpt stages. However, tomoGNN is faster than Blob and SBase, with relative total runtime reductions of 26% and 38%, respectively.

Overall, these denoised results indicate that our TomoGNN delivers an average (geo-mean) 18% improvement in TNS at the cost of a 27% runtime overhead, demonstrating clear PPA benefits over the default flow.

5.3 Ablation Studies

We now present ablation studies demonstrating the importance of multi-views in tomography, the effectiveness of our approach across different TCPs, the contributions of individual TomoGNN components, the benefit of our refinement strategy, and the benefit of soft guides over additional placement optimization.

Table 3: Benefits of additional runs.

Design	#Runs	rWL	Power	WNS	TNS	DRV
JPEG (NG45)	1	496	316	-137	-77.4	0
	10	497	315	-144	-74.8	0
	20	499	317	-134	-63.8	0
	30	498	317	-119	-70.8	0
CA53 (NG45)	1	11086	843	-56	-105.0	1369
	10	11194	845	-53	-70.2	1265
	20	11146	845	-55	-55.0	1220
	30	11121	844	-53	-56.2	1057
SweRV (GF12)	1	1.000	1.000	-0.250	-199	0
	10	0.999	0.993	-0.260	-183	0
	20	1.001	1.000	-0.250	-171	0
	30	0.999	0.993	-0.246	-158	0
Ariane (GF12)	1	1.001	1.006	-0.140	-148	0
	10	1.002	1.006	-0.136	-113	0
	20	1.001	1.006	-0.121	-96	0
	30	0.999	1.000	-0.123	-92	0

Table 4: Effectiveness of soft guides for different TCPs (ps).

Design	TCP	Type	rWL	Power	WNS	TNS	DRV
CA53 (NG45)	1100	Dflt	12203	1052	-95	-243.4	1910
		T-GNN	12151	1046	-97	-193.3	1502
	1200	Dflt	11149	917	-67	-178.6	1454
		T-GNN	11091	920	-65	-165.0	1787
	1300	Dflt	11137	843	-61	-89.7	784
		T-GNN	11121	844	-53	-56.2	1057
SweRV (GF12)	475	Dflt	1.000	1.000	-0.286	-204	0
		T-GNN	1.004	1.007	-0.259	-187	0
	500	Dflt	1.000	1.000	-0.244	-166	1
		T-GNN	1.004	1.000	-0.246	-158	0
	525	Dflt	1.000	1.000	-0.202	-98	0
		T-GNN	1.001	1.000	-0.202	-94	0

Benefits of additional runs. To demonstrate the benefits of additional runs, we generate different numbers of tomography features by running placeOpt on 1, 10, 20, and all 30 perturbed floorplans of the target design. This study is performed on JPEG (NG45), CA53 (NG45), Ariane (GF12) and SweRV (GF12) designs with post-routeOpt metrics reported in Table 3. We see that TNS generally improves with additional runs and plateaus around 20–30 runs. Additional runs beyond 30 do not bring further benefits in our experience. Based on this, we use tomography features from 30 runs in all of our experiments.

Benefits of soft guides across TCPs. To confirm effectiveness of our soft guides across TCPs, we generate soft guides for CA53 (NG45) and SweRV (GF12) with three different TCPs. To ensure post-routeOpt designs have WNS in the range of -50 to -100 ps, we tune both TCP and utilization. As a result, the floorplans change across different TCPs. As shown in Table 4, our soft guides achieve geo-mean improvements of 14.7% in TNS and 4.0% in WNS with similar power and slightly higher DRV numbers. This shows that TomoGNN consistently improves timing across a range of TCPs.

Benefits of TomoGNN model components. To isolate the contributions of the three key components of TomoGNN – FiLM modulation, dual-graph message passing, and the tomography calibration loss – we perform one-component-removal ablations on the JPEG and CA53 designs in both NG45 and GF12. The variant without dual-graph message passing (w/o *dual-MP*) applies three-layer GraphSAGE message passing solely on the netlist graph. All other components of the model and the evaluation flow remain the same. Experimental results are shown in Table 5. Removing FiLM or dual-graph message passing worsens TNS in all four cases, and removing the calibration loss worsens TNS in three of four cases; in the remaining CA53 (GF12) case, it slightly improves TNS but degrades DRV and runtime, indicating a worse overall tradeoff. Overall, the full TomoGNN gives the most consistent post-routeOpt

Table 5: Ablation study of TomoGNN.

Design	Type	rWL	Power	WNS	TNS	DRV	RT
JPEG (NG45)	T-GNN	498	317	-119	-70.8	0	0.63
	w/o FiLM	497	315	-112	-75.3	0	0.69
	w/o dual-MP	497	317	-158	-81.0	0	0.71
	w/o cal. loss	498	317	-145	-81.9	0	0.58
CA53 (NG45)	T-GNN	11121	844	-53	-56.2	1057	3.97
	w/o FiLM	11138	844	-59	-95.5	1392	4.01
	w/o dual-MP	11139	844	-57	-69.5	983	4.00
	w/o cal. loss	11109	843	-61	-87.3	1129	3.74
JPEG (GF12)	T-GNN	0.948	0.980	-0.325	-335	0	1.46
	w/o FiLM	0.940	0.988	-0.320	-397	0	1.57
	w/o dual-MP	0.963	0.996	-0.430	-413	0	1.45
	w/o cal. loss	0.929	0.986	-0.485	-488	0	1.49
CA53 (GF12)	T-GNN	1.000	0.999	-0.183	-162	1	4.88
	w/o FiLM	1.000	1.000	-0.223	-167	2	6.48
	w/o dual-MP	1.001	1.000	-0.182	-176	1	5.06
	w/o cal. loss	1.001	1.000	-0.175	-160	2	6.13

Table 6: Effectiveness of refinement strategy.

Design	Method	Type	rWL	Power	WNS	TNS	DRV	RT
CA53 (NG45)	Blob	No Refine	11137	844	-76	-115.3	1223	3.80
		Refine	11148	846	-49	-48.0	1391	5.81
	SBase	No Refine	11131	845	-64	-77.8	1269	4.05
		Refine	11109	844	-63	-77.4	1331	4.47
	T-GNN	No Refine	11154	845	-65	-66.8	1236	3.28
		Refine	11122	844	-53	-56.2	1057	3.97
Ariane (GF12)	Blob	No Refine	1.002	1.002	-0.146	-146	1	1.51
		Refine	1.002	1.006	-0.136	-133	0	3.79
	SBase	No Refine	1.002	1.004	-0.139	-117	0	1.51
		Refine	1.001	1.000	-0.128	-110	0	1.99
	T-GNN	No Refine	1.000	1.000	-0.156	-137	0	1.36
		Refine	0.998	0.994	-0.123	-92	0	1.71

timing improvement across designs and PDKs, confirming that all three components contribute to final timing quality.

Impact of soft guide refinement. We run our evaluation on the CA53 (NG45) and Ariane (GF12) designs, and compare results with and without refinement for Blob, SBase, and our TomoGNN method. Table 6 reports the post-routeOpt metrics for both settings. We observe that refinement consistently improves TNS (WNS), with geo-mean improvements of 38% (23%), 3% (5%) and 25% (20%) for Blob, SBase and TomoGNN, respectively. The additional runtime mainly comes from feature extraction from the Innovus DB, as the refinement step takes under 11 minutes for both designs. For CA53 (Ariane), refinement in the TomoGNN flow reduces the average cut on post-placeOpt critical paths from 8.91 (25.62) to 2.99 (14.99), explaining the observed PPA gains over runs without refinement.

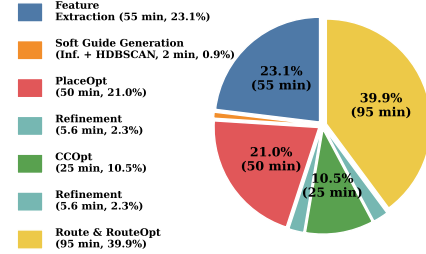
Benefit of soft guides over additional placement optimization. TomoGNN extracts features from multiple post-placeOpt runs. To evaluate the benefit of learned soft guides, we compare against Dflt-opt, a variant of Dflt that runs the placeOpt stage twice, with the second run using the incremental flag, before proceeding to downstream stages without using soft guides. Table 7 compares Dflt, Dflt-opt and T-GNN on BP and CA53 in both NG45 and GF12. Compared to Dflt-opt, T-GNN achieves geo-mean improvements of 10.8% and 16.9% in WNS and TNS, respectively. Since both flows use two placeOpt runs, the runtimes are very similar (Dflt-opt is slightly slower due to its incremental placeOpt loop). These results confirm that the PPA gains from T-GNN soft guides are not merely due to additional placeOpt effort.

5.4 Runtime Breakdown

Figure 5 decomposes the total wall time on CA53 (NG45) into feature extraction (i.e., tomography runs), soft guide generation (TomoGNN inference + HDBSCAN clustering), soft guide refinement across stages and the downstream P&R flow. Tomography feature extraction accounts for 23% of the total runtime, since the placeOpt runs are independent and executed in parallel. In addition, placeOpt itself is inexpensive compared to later stages such as routeOpt, making the overall cost of tomography runs low in practice. TomoGNN

Table 7: Comparison of Dflt, Dflt-opt and T-GNN.

Design	Type	rWL	Power	WNS	TNS	#DRV	RT
BP NG45	Dflt	25929	4470	-148	-321.9	0	5.28
	Dflt-opt	25759	4468	-136	-299	0	7.03
	T-GNN	25960	4467	-138	-288.3	0	6.96
CA53 NG45	Dflt	11137	843	-61	-89.7	784	2.68
	Dflt-opt	11087	845	-64	-72.4	426	3.37
	T-GNN	11121	844	-53	-56.2	1057	3.97
BP GF12	Dflt	1.000	1.000	-0.135	-405	0	6.45
	Dflt-opt	0.994	0.998	-0.147	-388	0	8.03
	T-GNN	1.002	1.000	-0.135	-265	0	7.22
CA53 GF12	Dflt	1.000	1.000	-0.213	-181	0	4.67
	Dflt-opt	0.999	0.999	-0.223	-174	1	5.67
	T-GNN	1.000	0.999	-0.183	-162	1	4.88

**Fig. 5: Runtime breakdown on CA53 (NG45).**

training is performed once per design and takes approximately two hours on CA53 (NG45). Since the synthesis netlist remains fixed during subsequent P&R explorations, the trained model is reused without retraining; we therefore exclude training time from the breakdown. Thus, the per-iteration P&R overhead comprises tomography feature extraction, model inference, soft guide generation and refinement, and the downstream P&R stages. Soft guide generation (GPU-based TomoGNN inference plus HDBSCAN) accounts for only 0.9% of the total runtime. Soft guide refinement runs after both the placeOpt and clockOpt stages; each refinement step uses only 2.3% of the total runtime, totaling 4.6%. The downstream P&R flow dominates: placeOpt accounts for 21%, clockOpt for 10.5% and routeOpt for 39.8%. Overall, our method adds modest overhead relative to the full P&R flow while achieving significant timing improvements under tight design constraints. Moreover, even when including the one-time training cost, the total wall time of our flow remains lower than that of the SBase baseline, which reflects how engineers typically form the soft guides.

6 Conclusion

We introduce *netlist tomography*, which generates multiple netlist views with timing, location and hierarchy features from multiple placement results. Building on these views, we propose *TomoGNN*, a self-supervised, tomography-aware graph model trained with PPA-driven contrastive loss; its embeddings are clustered using HDBSCAN to produce soft guides that are adaptively refined across P&R stages to track shifting critical paths. Experimental results in NG45 and GF12 show up to 42% and 37% improvements in WNS and TNS, respectively, with geometric-mean improvements of 12% and 18% over all testcases, compared to a leading commercial P&R tool flow. Directions for future work include incorporating DRV metrics into guide-selection and refinement objectives.

Acknowledgments

This work is supported by Apple; we thank Dr. Jiajia Li, Yifei Li, Lama Mouayad, Ala Qumsieh and Eric Smith for their constructive feedback. Our research on ML EDA is also broadly supported by the Samsung AI Center.

References

- [1] A. Agnesina, R. Liang, G. Pradipta, A. Rajaram and H. Ren, "GOALPlace: Begin with the End in Mind", *Proc. ACM/IEEE International Symposium on Physical Design (ISPD)*, 2025, pp. 2–10.
- [2] C. J. Alpert, A. B. Kahng, G.-J. Nam, S. Reda and P. Villarrubia, "A Semi-Persistent Clustering Technique for VLSI Circuit Placement", *Proc. ACM/IEEE International Symposium on Physical Design (ISPD)*, 2005, pp. 200–207.
- [3] G.-J. Nam, S. Reda, C. J. Alpert, P. G. Villarrubia and A. B. Kahng, "A Fast Hierarchical Quadratic Placement Algorithm", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25(4) (2006), pp. 678–691.
- [4] S. Bai, F. Zhang and P. H. S. Torr, "Hypergraph Convolution and Hypergraph Attention", *Pattern Recognition* 110 (2021), 107637.
- [5] I. Bustany, G. Gasparyan, A. B. Kahng, Y. Koutis, B. Pramanik and Z. Wang, "An Open-Source Constraints-Driven General Partitioning Multi-Tool for VLSI Physical Design", *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2023, pp. 1–9.
- [6] T.-B. Chan, A. B. Kahng and M. Woo, "Revisiting Inherent Noise Floors for Interconnect Prediction", *Proc. ACM/IEEE International Workshop on System-Level Interconnect: Problems and Pathfinding (SLIP)*, 2020, pp. 1–7.
- [7] Z. Chen, C.-K. Cheng, Z. Song and Y. Wang, "Noise-Aware Circuit Clustering Based on Analytical Placement Evolution", *Proc. ACM/IEEE Workshop on System-Level Interconnect Pathfinding (SLIP)*, 2024, pp. 1–9.
- [8] C.-K. Cheng, A. B. Kahng, S. Kundu, Y. Wang and Z. Wang, "Assessment of Reinforcement Learning for Macro Placement", *Proc. ACM/IEEE International Symposium on Physical Design (ISPD)*, 2023, pp. 158–166.
- [9] C.-K. Cheng, A. B. Kahng, S. Kundu, Y. Wang and Z. Wang, "An Updated Assessment of Reinforcement Learning for Macro Placement", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 45(8) (2026), pp. 3654–3668.
- [10] Y. Cheon and D. F. Wong, "Design Hierarchy Guided Multilevel Circuit Partitioning", *Proc. ACM/IEEE International Symposium on Physical Design (ISPD)*, 2002, pp. 30–35.
- [11] W. Ding, Z. Zhang, G. He and P. Cao, "A Physical and Timing Aware Placement Optimization Framework Based on Graph Neural Network", *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2024, pp. 1–9.
- [12] M. Fogaça, A. B. Kahng, E. Monteiro, R. Reis, L. Wang and M. Woo, "On the Superiority of Modularity-Based Clustering for Determining Placement-Relevant Clusters", *Integration: The VLSI Journal* 74 (2020), pp. 32–44.
- [13] W. L. Hamilton, R. Ying and J. Leskovec, "Inductive Representation Learning on Large Graphs", *Proc. International Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 1024–1034.
- [14] B. Hu and M. Marek-Sadowska, "Fine Granularity Clustering-Based Placement", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 23(4) (2004), pp. 527–536.
- [15] J. Jeong and T. Kim, "Machine Learning Driven Early Clustering for Multi-bit Flip-Flop Allocation", *Proc. ACM/IEEE Symposium on Machine Learning for CAD (MLCAD)*, 2025, pp. 1–7.
- [16] K. Jeong and A. B. Kahng, "Methodology From Chaos in IC Implementation", *Proc. International Symposium on Quality Electronic Design (ISQED)*, 2010, pp. 885–892.
- [17] A. B. Kahng, S. Kang, S. Kundu, K. Min, S. Park and B. Pramanik, "PPA-Relevant Clustering-Driven Placement for Large-Scale VLSI Designs", *Proc. ACM/IEEE Design Automation Conference (DAC)*, 2024, pp. 1–6.
- [18] A. B. Kahng, S. Kundu and D. Yoon, "Placement Tomography-Based Routing Blockage Generation for DRV Hotspot Mitigation", *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2024, pp. 1–9.
- [19] A. B. Kahng, R. Varadarajan and Z. Wang, "Hier-RTLMP: A Hierarchical Automatic Macro Placer for Large-Scale Complex IP Blocks", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43(5) (2024), pp. 1552–1565.
- [20] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization", *Proc. International Conference on Learning Representations (ICLR)*, 2019.
- [21] Y.-C. Lu, H.-H. Hsiao, R. Liang, W.-H. Liu and H. Ren, "C3PO: Commercial-Quality Global Placement via Coherent, Concurrent Timing, Routability, and Wirelength Optimization", *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2026, pp. 765–771.
- [22] Y.-C. Lu, S. Pentapati and S. K. Lim, "The Law of Attraction: Affinity-Aware Placement Optimization Using Graph Neural Networks", *Proc. ACM/IEEE International Symposium on Physical Design (ISPD)*, 2021, pp. 7–14.
- [23] Y.-C. Lu, T. Yang, S. K. Lim and H. Ren, "Placement Optimization via PPA-Directed Graph Clustering", *Proc. ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, 2022, pp. 1–6.
- [24] L. McInnes, J. Healy and S. Astels, "hdbscan: Hierarchical Density Based Clustering", *Journal of Open Source Software* 2(11) (2017), 205.
- [25] K. Min, S. Kwon, S.-Y. Lee, D. Kim, S. Park and S. Kang, "ClusterNet: Routing Congestion Prediction and Optimization Using Netlist Clustering and Graph Neural Networks", *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2023, pp. 1–9.
- [26] C. Morchdi, C.-H. Chiu, W. L. Lee, T.-W. Huang and Y. Zhou, "Enhancing Graph Partitioning with Reinforcement Learning-based Initialization", *Proc. IEEE High Performance Extreme Computing Conference (HPEC)*, 2025, pp. 1–8.
- [27] A. van den Oord, Y. Li and O. Vinyals, "Representation Learning with Contrastive Predictive Coding", *arXiv preprint arXiv:1807.03748*, 2018.
- [28] E. Perez, F. Strub, H. de Vries, V. Dumoulin and A. C. Courville, "FiLM: Visual Reasoning with a General Conditioning Layer", *Proc. AAAI Conference on Artificial Intelligence* 32(1) (2018), pp. 3942–3951.
- [29] V. A. Traag, L. Waltman and N. J. van Eck, "From Louvain to Leiden: Guaranteeing Well-Connected Communities", *Scientific Reports* 9 (2019), 5233.
- [30] N. Viswanathan, M. Pan and C. Chu, "FastPlace 3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control", *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2007, pp. 135–140.
- [31] Z. Wang, W. Zhao, Y. Pu, L. Chen, W. W. K. Thong, W. Sheng, T.-Y. Ho and B. Yu, "ParSGCN: Bridging the Gap Between Emulation Partitioning and Scheduling", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 44(3) (2025), pp. 1180–1192.
- [32] J. Z. Yan, C. Chu and W.-K. Mak, "SafeChoice: A Novel Approach to Hypergraph Clustering for Wirelength-Driven Placement", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30(7) (2011), pp. 1020–1033.
- [33] J. Zhang and Y. Luo, "Degree Centrality, Betweenness Centrality, and Closeness Centrality in Social Network", *Proc. 2nd International Conference on Modelling, Simulation and Applied Mathematics (MSAM)*, 2017, pp. 300–303.
- [34] BlobPlace GitHub. <https://github.com/ABKGroup/BlobPlacement>
- [35] NanGate45 PDK. <https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts/tree/master/flow/platforms/nangate45>
- [36] Netlist Tomography repository. <https://github.com/ABKGroup/NetlistTomography>
- [37] OpenROAD-flow-scripts. <https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts>
- [38] TILOS MacroPlacement repository. <https://www.github.com/TILOS-AI-Institute/MacroPlacement>