PARoute2: Enhanced Analog Routing via Performance-Drive Guidance Generation

Peng Xu, Jindong Tu, Guojin Chen, Keren Zhu, *Member, IEEE*, Tinghuan Chen, *Member, IEEE*, Tsung-Yi Ho, *Fellow, IEEE*, and Bei Yu, *Senior Member, IEEE*

***Abstract*—Analog routing is crucial for performance optimization in analog circuit design, but conventionally takes significant development time and requires design expertise. Recent research has attempted to use machine learning (ML) to generate guidance to preserve circuit performance after analog routing. These methods face challenges such as expensive data acquisition and biased guidance. This article presents AnalogFold, a new paradigm of analog routing that leverages ML to provide performance-oriented routing guidance. Our approach learns performance-driven routing guidance and uses it to help automatic routers for performance-driven routing optimization. We propose to use a 3DGNN that incorporates cost-aware distance to make accurate predictions on post-layout performance. A pool-assisted potential relaxation process derives the effective routing guidance. The experimental results on multiple benchmarks under the TSMC 40 nm technology node demonstrate the superiority of the proposed framework compared to the cutting-edge works.**

***Index Terms*—CADCAM, design automation, electronic design automation and methodology.**

I. Introduction

PRESERVING good performance in analog circuit design requires effective analog routing [1]. However, achieving a satisfactory circuit routing solution is a time-consuming and expertise-intensive task. Unlike digital circuits, analog circuits are extremely vulnerable to layout parasitics and coupling, making traditional analog routing heavily reliant on the experience of seasoned experts. Despite ongoing endeavors to automate analog circuit routing, its practical implementation in design flows has been limited. This limitation can be attributed to the challenges associated with capturing designers' expertise and accommodating the diverse array of layout-dependent effects [2].

Early analog routing methods employing expert-designed experience can be classified int three primary categories: 1) template-based [3], [4]; 2) simulation-based [5], [6]; and 3) heuristic constraint-based approaches [7], [8], [9], [10]. Template-based methods and simulation-based methods often struggle to handle complex designs due to their lack of flexibility and time-consuming nature. Heuristic constraint-based methods employ carefully designed heuristic principles to guide the routing decision process. The routing decisions are made based on these heuristics to achieve a satisfactory solution. The symmetric net pair constraint is the most commonly used heuristic in routing [7], [11], [12], [13], [14]. Ou et al. [7] expanded upon this constraint by introducing different levels of geometric matching constraints. Other approaches include prohibiting routing over the active regions of transistors [8], optimizing power routing [9], and reducing wire load [10]. However, when dealing with complex analog designs, these simple heuristics often fall short in addressing the diverse range of layout-dependent effects or incorporating the expertise of design professionals.

Machine learning (ML) methods summarize the experience from existing routing solutions, but suffering from scarcity of manual data [15] and lack of explicit performance consideration [16]. GeniusRoute [15] is a novel analog routing paradigm that has used generative neural networks to provide routing guidance. By employing automatic routing guidance, it mimics manual routing patterns and hence incorporates design experience into an automated engine, resulting in enhanced routing performance. However, the human imitation methodology has several issues. The training process relies on manual labeling data and limits the model's capability. Moreover, the existing methods do not explicitly consider performance optimization and lead to biased routing guidance. This hurts the generality of the routing guidance and creates a gap between the guidance and analog circuit performance.

To address these challenges, we introduce a performance-driven analog routing method that learns routing guidance from an automated routing engine. One of the most challenging aspects lies in the performance modeling of routing guidance. In comparison to placement benefits from existing performance models [17], analog routing guidance generation faces challenges related to discreteness, sparsity, and relatively low resolution in 3-D space. Considering the influence of parasitic parameters, even minor distorted predictions in guidance can result in substantial performance discrepancies. Inspired by recent advancements in protein structure prediction [18], [19], we propose the **PARoute** framework, which utilizes a gradient descent-based routing guidance

Received 6 October 2024; revised 16 January 2025; accepted 5 March 2025. Date of publication 11 March 2025; date of current version 22 September 2025. This work was supported in part by the Research Grants Council of Hong Kong, SAR, under Grant RFS2425-4S02, Grant CUHK14211824, and Grant CUHK14201624; in part by the Natural Science Foundation of Beijing, China, under Grant Z230002; and in part by the National Natural Science Foundation of China under Grant 62304197. This article was recommended by Associate Editor G. G. E. Gielen. *(Corresponding author: Bei Yu.)*

Peng Xu, Guojin Chen, Tsung-Yi Ho, and Bei Yu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, SAR (e-mail: byu@cse.cuhk.edu.hk).

Jindong Tu and Tinghuan Chen are with the School of Science and Engineering, The Chinese University of Hong Kong (Shenzhen), Shenzhen 518172, Guangdong, China.

Keren Zhu is with the School of Microelectronics and State Key Laboratory of Integrated Chips and Systems, Fudan University, Shanghai 200433, China.

Digital Object Identifier 10.1109/TCAD.2025.3550445

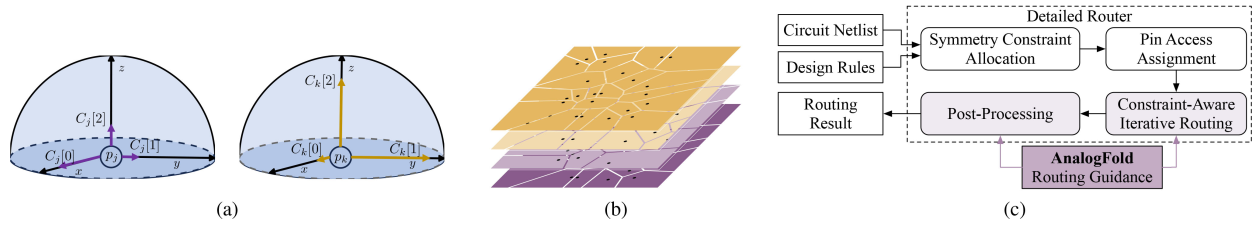


Fig. 1. (a) Two examples of nonuniform routing guidance. (b) 3-D visualization of the nonuniform routing guidance, with each point as a pin access. (c) Overall flow of our performance-driven analog routing method.

relaxation method to optimize routing guidance for individual nets instead of generating a complete routing guidance map. However, this leads to substantial GPU memory usage and considerable overhead in inference time. We propose a performance-driven routing guidance generation method that enables us to efficiently execute routing guidance generation tasks using a few GPU resources while achieving faster speeds. The main contributions of this article are listed as follows.

1) We introduce a performance-driven analog routing approach, which learns from the automatically generated routing patterns using their performance metrics.
2) A nonuniform routing guidance is proposed to effectively address sparsity issues by assigning routing guidance to different nets.
3) We proposed a customized PARoute framework to enable accurate modeling of the performance potential of routing guidance. PARoute contains a heterogeneous routing graph, a protein-inspired 3DGNN network, and a pool-aided potential relaxation process.
4) We further propose using a 3DGRU network-based routing guidance generation process and incorporating a customized routing order to determine the generation sequence. Additionally, we have integrated the selection process in the generated results with the performance prediction model.
5) The experimental results under the TSMC 40nm technology node demonstrate a significant improvement of the proposed framework compared to the cutting-edge works, with up to 3671.00 μV, 30.33 dB, 169.2 MHz, 38.141 dB, and 2028 μV_{rms} improvement in Offset Voltage (OV), Common-ode Rejection Ratio (CMRR), BandWidth, DC Gain, and noise metrics.

II. Preliminary and Motivation

A. Analog Routing Problem Formulation

Analog routing mainly involves placed modules, nets, self-symmetry nets, symmetry net pairs, and design rules. The objective is to route all the nets and optimize post-layout performance. They are defined as follows.

Definition 1 (Placed Modules): Let $\mathcal{M} = \{m_i | 1 \leq i \leq |M|\}$ be the collection of placed devices with a layout.

Definition 2 (Nets): A net N is defined as a collection of pins that require interconnection. Let $\mathcal{N} = \{n_i | 1 \leq i \leq |\mathcal{N}|\}$ be the set of Nets.

Definition 3 (Pin Access Point): Pin access points refer to the intersections between pin geometry and routing grids. Each pin has at least one access point.

Definition 4 (Self-Symmetry Nets): A self-symmetry net is characterized by its symmetry about a specific axis or point. Let $\mathcal{N}^{SS} = \{n_i^{SS} \mid 1 \leq i \leq |\mathcal{N}^{SS}|\}$ be the set of self-symmetry nets.

Definition 5 (Symmetry Net Pairs): A symmetry net pair consists of two nets N_1 and N_2 that exhibit symmetry concerning a defined axis or point. We define the group of symmetry net pairs as $\mathcal{N}^{SP} = \{n_i^{SP} \mid 1 \leq i \leq |\mathcal{N}^{SP}|\}$.

To achieve good manufacturability, analog routing needs to follow the design rules. Design rules are represented as a set of constraints that must be adhered to during the routing process. These rules encompass various requirements such as spacing, width, and layer constraints.

For efficiency and flexibility, state-of-the-art analog detailed routers route on tracks without preferred direction of each metal layer [15], [20]. These routing tracks form a 3-D grid graph $G = (V, E)$, where V represents grid points (intersection points of the tracks) and E denotes edges. An edge $e = (u, v) \in E$ indicates the connection from vertex u to v. Edges between neighboring vertices can be horizontal or vertical wire segments for intralayer connections or vias for interlayer connections.

Definition 6 (3-D Routing Grid): The variable x denotes the horizontal coordinate within the 3-D routing grid, y represents the vertical coordinate, and z corresponds to the metal layer of the grid.

The z dimension indicates the different metal layers used in the designs. Foundries provide different metal options for each technology node determined by factors, such as material composition, width, spacing, thickness, and other specifications.

In this work, we attempt to use ML models to generate effective guidance for analog routing. According to the definitions, we define the guided analog routing problem as follows.

Problem 1 (Guided Analog Routing): Guided analog routing takes placement $\mathcal{M}$, nets $\mathcal{N}$, a set of self-symmetry nets $\mathcal{N}^{SS}$, a group of symmetry net pairs $\mathcal{N}^{SP}$, design rules, and a set of generated nonuniform routing guidance $\mathcal{C} = \{\boldsymbol{C}_i | 1 \leq i \leq |N|\}$, as input, and routes all the nets while honoring symmetric constraints and routing guidance with optimal post-layout performance.

B. Theoretical Analysis Between Layout and Performance

The layout's influence on circuit performance is predominantly due to interconnect parasitic effects and device mismatches [21]. The early work on analog layout attempted to employ sensitivity analysis to directly measure these layout-induced effects on interconnect and mismatches parasitic effects [22], [23].

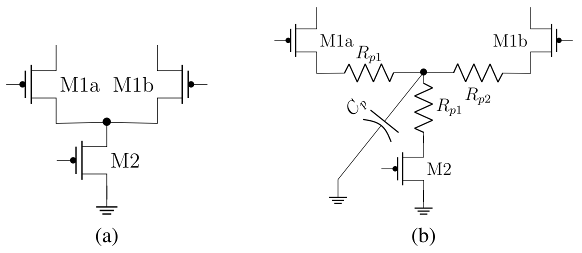


Fig. 2. Modeling of interconnect parasitic. (a) Differential pair without parasitic. (b) Differential pair with parasitic capacitor and resistors.

For the interconnect parasitic effect, the interconnect parasitics are modeled as a star circuit with resistors and a capacitor, as shown in Fig. 2. Both sheet capacitance and sheet resistance correlate with the primary routing layer z_{ap} at each access point. The wire segment resistances $R_{p,i}(x_{ap}, y_{ap}, z_{ap})$ are expressed in terms of layout parameters:$R_{p,i}(x_{ap}, y_{ap}, z_{ap}) = [\rho_{\text{square}}\, \text{WL}_{\text{wire}}(x_{ap}, y_{ap}, z_{ap})/n\, \text{WL}_{\text{wire},k}(x_{ap}, y_{ap}, z_{ap})]$, where ρ_{square} is the sheet resistance and $\text{WL}_{\text{wire},k}$ is the wire length of the kth wire segment. The total parasitic capacitance could be represented by functions of the pin access points coordinates $C_{p,i}(x_{ap}, y_{ap}, z_{ap})$. We then calculate the performance degradation $\Delta P_{t,net}$ as

$$\Delta P_{t,net}(x_{ap}, y_{ap}, z_{ap}) = \sum_{k=1}^{m} \Big(S^{j}_{C_{p,k}} C_{p,k}(x_{ap}, y_{ap}, z_{ap}) + \sum_{i=1}^{n_k} S^{j}_{R_{p,ki}} R_{p,ki}(x_{ap}, y_{ap}, z_{ap}) \Big) \quad (1)$$

where m is the number of nodes minus the ground node and n_k is the number of terminals of net k. $S^{j}_{C_{p,k}} = [(\Delta P_t)/(\delta C_{p,k})]$ and $S^{j}_{R_{p,ki}} = [(\Delta P_t)/(\delta R_{p,ki})]$ are the sensitivities of performance metric P_t to small changes in the parasitic capacitance $C_{p,k}$ and the parasitic resistances $R_{p,ki}$.

For device mismatch, the mismatch of MOS transistors can be estimated using variances for threshold voltage (V_{T0}) and transconductance parameter (β) as in [22]. Using sensitivity information, the performance degradation concerning the device mismatch effect can be estimated

$$\Delta P_{t,\text{module}} = \sum_{i=1}^{m} \Big[S^{j}_{\Delta V_{T0,i}} \cdot 3\sigma(V_{T0}) + S^{j}_{\Delta \beta_i} \cdot 3\sigma(\beta) \Big] \quad (2)$$

where $S^{j}_{\Delta V_{T0,i}} = [(\delta P_t)/(\delta \Delta V_{T0,k})]$ and $S^{j}_{\Delta \beta_i} = [(\delta P_t)/ (\delta \Delta \beta_k)]$ are the sensitivities of performance metric P_t to small changes in V_{T0} and β. The equations for these variances are derived with constants and variables reflecting process and layout specifics, as: $\sigma^2(V_{T0}) = (A^2_{V_{T0}}/\text{Area}) + S^2_{V_{T0}} D(x_m, y_m)^2$ and $\sigma^2(\beta) = (A^2_{\beta}/\text{Area}) + S^2_{\beta} D(x_m, y_m)^2$. (x_m, y_m) is coordinates of the modules of targeted analog circuit, $D(\cdot, \cdot)$ is the distances between them.

Finally, the relationship between performance degradation and the layout can be simplified as the sum of the effects brought by the device mismatches and the interconnect parasitic effects

$$\Delta P_t = \Delta P_{t,net}(x_{ap}, y_{ap}, z_{ap}) + \Delta P_{t,\text{mudule}}(x_m, y_m). \quad (3)$$

However, the linear approximation model constrains its effectiveness in addressing the nonlinear behavior of analog circuits. Additionally, determining these sensitivities is extremely time-consuming, which further limits scalability. Thus, We designed a customized graph neural network to predict the post-layout performance instead in Section IV-B, honoring the input features of the early performance approximation model.

C. Advanced Protein Structure Prediction

In recent years, deep-learning-based approaches have shown great promise in predicting protein structures [18], [19]. These approaches have been used to predict various aspects of protein structure, including contact maps, real-valued distances, and quality assessment. A key issue is correctly predicting the 3-D shapes of some molecules from well-studied protein families. Different types of relative 3-D information can be derived from 3-D molecular graphs, and they can be important in molecular learning, such as bond lengths, and angles between bonds [24], [25]. Inspired by recent ML advancements in protein structure prediction [18], [19], [25], [26], we have noticed that routing guidance generation and protein structure prediction both face challenges related to discreteness, sparsity, and relatively low resolution in 3-D space. We proposed the 3DGNN and 3DGRU networks, that are customized for the modeling of analog routing.

III. Performance-Driven Analog Route

The existing analog routing algorithms only consider summarizing human experience and fail to guarantee performance improvement. To tackle the performance-driven analog routing problem, PARoute divides it into two subproblems: 1) nonuniform routing guidance generation and 2) guided analog detailed routing.

A. Nonuniform Routing Guidance Generation

GeniusRoute uses a uniform 2-D routing guidance map, which has significant challenges. The model's performance may be largely compromised when handling designs of varying sizes or aspect ratios. More importantly, uniform 2-D guidance fails to handle the sparsity issue of analog layout that might assign two closed pins to the same pixel, hindering the modeling of resource competition among different pins.

To address these issues, we propose a nonuniform adaptive cost guidance that assigns each pin a different cost guidance, as shown in Fig. 1(a). Essentially, the routing guidance is a cost guidance field that indicates the priority of a given net being routed. In this case, the routing guidance for p_k in the x direction is smaller than that for p_j. Therefore, it encourages p_k to route the wires horizontally. The resulting priority regions predicted by the performance model are shown in Fig. 1(b). This approach effectively addresses sparsity issues in the route layout by cost guidance to different nets and inherently supports a 3-D cost map.

B. Symmetry Constraint Allocation and Employment

Our analog routing engine employs an automated symmetric constraint extraction method proposed in [20], to identify and

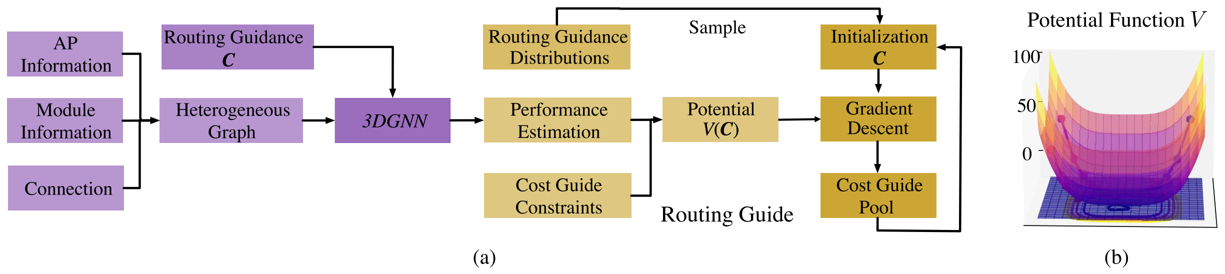


Fig. 3. (a) Overview of our PARoute, which trains a 3DGNN to model the potential function of routing guidance and uses relaxation to derive the optimized routing guidance. (b) Optimize routing guidance via potential relaxation.

assign suitable symmetry constraints to matched nets based on the geometric locations of their pins.

Once the placement layout and routing netlist are acquired, we perform a symmetry constraint allocation procedure to determine the appropriate constraints for matched nets using a symmetry-weighted graph. This process begins by constructing an undirected graph $\mathcal{G}_{\text{sym}} = (\mathcal{V}_{\text{sym}}, \mathcal{E}_{\text{sym}})$, where $\mathcal{V}_{\text{sym}}$ represents a set of vertices of nets, and $\mathcal{E}_{\text{sym}}$ represents a set of weighted edges. For each net n_i in the circuit netlist, two separate vertices, v and v', are added to $\mathcal{V}_{\text{sym}}$. The symmetry weight of each edge is then computed based on the estimated degree of symmetry or self-symmetry of the nets.[1]

We first compute the estimated degree of self-symmetry sym_i and the correlated symmetry axis λ_i for each net n_i

$$\text{sym}_i = f_i(\lambda_i),\ \lambda_i = \underset{1\le k,k'\le|P_i|}{\text{argmax}}\ f_i\left(\frac{x_{i,k}+x_{i,k'}}{2}\right)$$
$$f_i(x) = \frac{\sum_k a_{i,k}(x)}{|P_i|} \tag{4}$$

where $a_{i,k}(x)$ is a function with output 1 if $p_{i,l}$ has a corresponding symmetric pin in the set of pins P_i in n_i with respect to x, and 0 otherwise.

An edge $(v_i, v_i', \text{sym}_i)$ is added to $\mathcal{E}_{\text{sym}}$ if $\text{sym}_i > 0$. Otherwise, we compute the estimated degree of symmetry $\text{sym}_{i,j}$ and the correlated symmetry axis $\lambda_{i,j}$ as follows:

$$\text{sym}_{i,j} = f_{i,j}(\lambda_{i,j}),\ \lambda_{i,j} = \underset{1\le k,k'\le|P_j|}{\text{argmax}}\ f_{i,j}\left(\frac{x_{i,k}+x_{j,k'}}{2}\right)$$
$$f_{i,j}(x) = \frac{\max\left(\sum_k a^j_{i,k}(x), \sum_{k'} a^i_{j,k'}(x)\right)}{\max\left(|P_i|, |P_j|\right)}. \tag{5}$$

Similarly, we add an edge $(v_i, v_j, 2\text{sym}_{i,j})$ to $\mathcal{E}_{\text{sym}}$ if $\text{sym}_{i,j} > 0$. After constructing the graph, we apply Edmonds' Blossom algorithm [27] to find the maximum-weight matching. Then, we translate the matching result to specify the symmetry constraints for nets.

As for the symmetry employment, we utilize an extended mirroring technique to route symmetric nets by combining obstacles on both sides of the symmetry axis [28]. The A* search algorithm is used for obstacle-aware pathfinding for unrouted pin clusters. Once a route is found, it is mirrored to produce a symmetric solution. For partial-symmetric nets, an additional pathfinding step connects the remaining pins and clusters.

[1]For simplicity, we focus on the computation of horizontal symmetry in this description, noting that vertical symmetry can be achieved using the same approach.

C. Guided Analog Routing

As shown in Fig. 1(c), PARoute leverages the routing guidance $\boldsymbol{C}$ generated by ML models to make routing decisions. Our analog routing flow mainly consists of two steps.

1) Detailed routing routes all nets via constraint-aware iterative routing as in [20], honoring design rules, symmetric constraints, and input nonuniform routing guidance. During each iteration, routing guidance $\boldsymbol{C}$ are honored via penalties in the cost function along different directions for different pin access points.
2) In post-processing, the routing solution will be refined for the remaining design rule violations [15].

IV. Performance-Driven Routing Guidance Generation

In this section, we first establish the framework and definition of the problem for generating performance-driven routing guidance. Subsequently, we used a 3DGNN as the classifier to predict the potential of routing guidance and a 3DGRU (graph recurrent neural network) as the generator network to produce diverse routing guidances, respectively. Finally, we elaborated on how to jointly satisfy the diversity requirements and maximize the given Figure of Merit (FoM) of routing guidance through a carefully designed Loss function.

Fig. 3(a) and (b) shows the overall flow of our paroute algorithms. Fig. 4 shows the overall flow of our routing guidance generation framework. Our Analog framework differs from traditional analog methods that rely on summarizing and learning from human analog routing solutions. Instead, we use a graph generation model to produce diverse routing guidance and explicitly construct a performance prediction model to select the routing guidance.

A. Overview

PARoute2 employs a generator to create diverse routing guidances, complemented by a discriminator that predicts simulation outcomes to refine the routing guidance generation process according to specified criteria. Our approach begins by selecting high-quality routing guidance samples

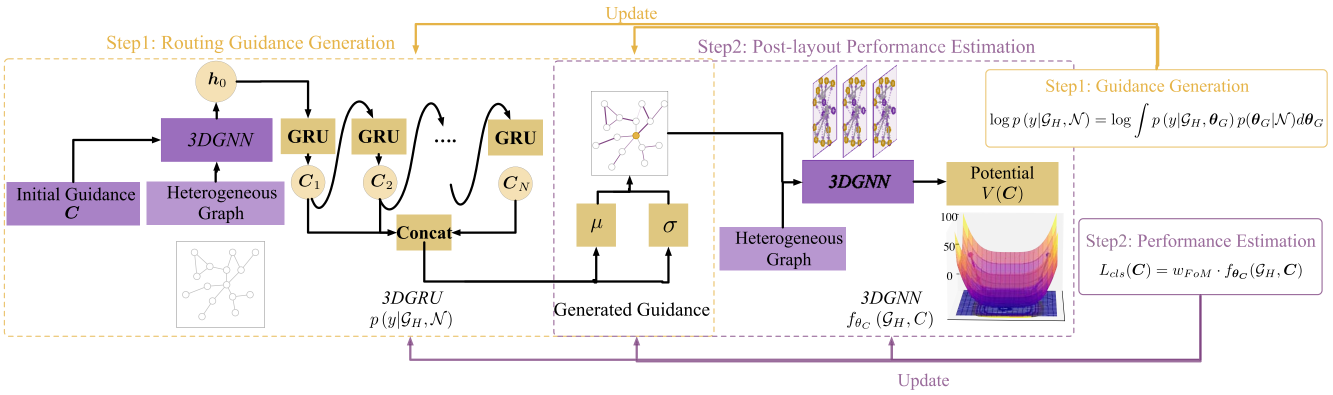

Update
Step1: Routing Guidance Generation
Step2: Post-layout Performance Estimation
Step1: Guidance Generation
$\log p(y|\mathcal{G}_H,\mathcal{N}) = \log \int p(y|\mathcal{G}_H,\boldsymbol{\theta}_G)\,p(\boldsymbol{\theta}_G|\mathcal{N})d\boldsymbol{\theta}_G$
h_0
3DGNN
GRU
GRU
GRU
C_1
C_2
C_N
Initial Guidance $\boldsymbol{C}$
Heterogeneous Graph
Concat
μ
σ
3DGRU $p(y|\mathcal{G}_H,\mathcal{N})$
Generated Guidance
Heterogeneous Graph
3DGNN
Potential $V(\boldsymbol{C})$
3DGNN $f_{\theta_C}(\mathcal{G}_H,C)$
Step2: Performance Estimation
$L_{cls}(\boldsymbol{C}) = w_{FoM}\cdot f_{\theta_C}(\mathcal{G}_H,\boldsymbol{C})$
Update

Fig. 4. Overview of our PARoute2 framework. First, the routing guidance generation model is trained to maximize the KL divergence. Second, the post-layout prediction model is utilized to select the routing guidance with high potential, and optimize the generator as well.

through nondominated sorting of simulation results from various routing outcomes, which are then used to construct the training dataset. Following this, we utilize a 3DGRU-based generator, customized with a Sequential Routing Order, to generate the routing guidance. Finally, we train the 3DGRU-based generator using a hybrid loss that integrates both generative and discriminative objectives, while a 3DGNN-based classifier guides the final routing guidance generation process.

Problem Formulation: We propose a Bayesian-based model for routing guidance generation. In our routing guidance generation task, we model the parameter θ_G of the routing guidance generator as a distribution to enhance flexibility. An effective routing guidance predictor should be generated based on the input routing graph and the distribution of routing guidance choices for edges. Following established practices in Bayesian ML, we can express our learning objective as:

$$L_{\text{gen}}(\boldsymbol{C}|\mathcal{G}_H, \mathcal{N}) = \log p(\boldsymbol{C}|\mathcal{G}_H, \mathcal{N}). \tag{6}$$

The optimization of the performance-driven selection process is based on the FoM results of the predicted performance. Given a target specification, the discriminative model $f_m(\cdot)$ with parameters θ_M, the performance-driven selection problem can be formulated as

$$L_{\text{cls}}(\boldsymbol{C}) = \boldsymbol{w}_{\text{FoM}} \cdot f_{\boldsymbol{\theta}_C}(\mathcal{G}_H, \boldsymbol{C}) \tag{7}$$

where $\boldsymbol{w}_{\text{FoM}}$ represents the weight preferences for different metrics corresponding to a given spec.

In this co-evolution problem, the routing guidance generation and performance-driven selection can be viewed as a bi-level optimization challenge. The routing guidance generation model aims to produce diverse routing guidance that may negatively impact post-layout performance. Conversely, the routing guidance classifier's goal is to identify the most promising options from the generated routing guidance. In other words, the routing guidance generation model seeks to maximize the Kullback–Leibler divergence, while the routing guidance classifier model aims to maximize the post-layout FoM under the specified criteria

$$\begin{cases} \boldsymbol{C}^* = \underset{\boldsymbol{C}}{\operatorname{argmax}} L_{\text{cls}}\left(\boldsymbol{\theta}_G^*(\boldsymbol{C}), \boldsymbol{C}\right) \\ \boldsymbol{\theta}_G^* = \underset{\boldsymbol{\theta}_G}{\operatorname{argmin}} L_{\text{gen}}(\boldsymbol{\theta}_G, \boldsymbol{C}) \end{cases} \tag{8}$$

where L_{cls} is the loss function of performance predictor and L_{gen} is the loss function of routing guidance generator.

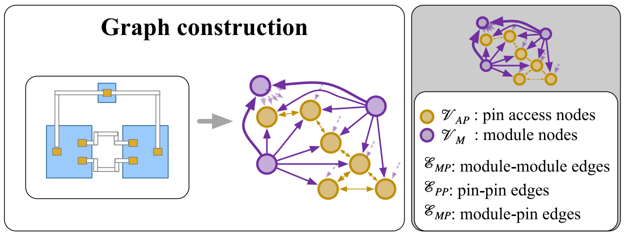

Graph construction
$\mathscr{V}_{AP}$: pin access nodes
$\mathscr{V}_M$: module nodes
$\mathscr{E}_{MP}$: module-module edges
$\mathscr{E}_{PP}$: pin-pin edges
$\mathscr{E}_{MP}$: module-pin edges

Fig. 5. Construction of heterogeneous graph for modeling analog routing.

B. Heterogeneous Graph for Analog Routing

Routing cost maps for global routing can be formulated into graphs naturally since connections of pin access points and modules form them. To model routing cost maps for analog global routing problems, we propose a heterogeneous graph representation for improved accuracy, efficiency, and complexity. We design a heterogeneous graph $\mathcal{G}_H = <\mathcal{V}_{AP}, \mathcal{V}_M, \mathcal{E}_{PP}, \mathcal{E}_{MP}, \mathcal{E}_{MM}>$ to represent the interactions between pin access points and modules, as shown in Fig. 5.

The vertex sets $\mathcal{V}_{AP}$ and $\mathcal{V}_M$ correspond to the pin access points and modules, respectively. $\mathcal{E}_{PP}$ is designed to reflect the interplay between different pin access points. $\mathcal{E}_{MM}$ contains the edges that connect the modules according to the netlist. The edges $\mathcal{E}_{PM}$ to model the relationship between the modules and the pin access points.

Point-to-Point Connections: If two pin access points $v_{AP_i}, v_{AP_j} \in \mathcal{V}_{AP}$ are connected by a segment or wire, we add an edge (v_{AP_i}, v_{AP_j}) to $\mathcal{E}_{PP}$. As shown in Fig. 5, these edges indicate the physical connection relationships between pin access points. According to (1), the input features of pin access points are correlated with post-layout performance in interconnect parasitic effects. The most critical features correspond to the 3-D coordinate information of the access

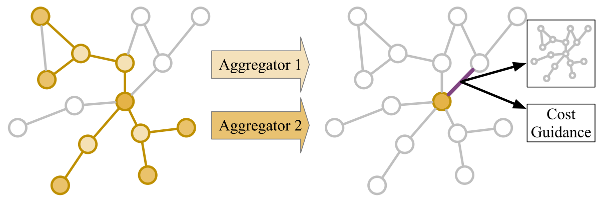


Fig. 6. We cast the routing guidance prediction into a link prediction task.

Algorithm 1 Cost-Aware Message Passing for 3DGNN

```
1: procedure CAMP
2:     $e_k^l \leftarrow \phi^e\left(e_k, v_{r_k}, v_{s_k}, \mathcal{E}_{s_k}, \rho^{p\rightarrow e}\left(\{r_h\}_{h=r_k\cup s_k}\right)\right)$;    ▷ Update
3:     $v_i^l \leftarrow \phi^v\left(v_i, \rho^{e\rightarrow v}\left(\mathcal{E}_i^l\right)\right)$;    ▷ Aggregate
4:     $u^l \leftarrow \phi^u\left(u, \rho^{v\rightarrow u}\left(\mathcal{V}^l\right)\right)$;    ▷ Combine
5: end procedure
```

point itself, including its corresponding coordinates (x_{ap}, y_{ap}) and the primary routing layer z_{ap} used. This coordinate information, along with the pin index p_i that the pin access point belongs to, and the interconnection information between pin access points, ultimately reflects the parasitic resistance $R_{p,i}$ and parasitic capacitance C_p. We use these features to attempt to reflect the impact of layout on post-layout performance with the interconnect parasitic effects.

Module-to-Module Connections: Each vertex in $\mathcal{V}_M$ represents a module on the analog layout. For vertices $v_{M_i}, v_{M_j} \in \mathcal{V}_M$, if a module M_i and a module M_j are connected by a net, we add an edge (v_{M_i}, v_{M_j}) to $\mathcal{E}_{MM}$. As a result, $\mathcal{E}_{MM}$ can reflect the logical connections between modules. According to (2), module input features correlate with post-layout performance through mismatch effects. The most significant features include the module's device type and spatial characteristics—specifically its coordinates (x_m, y_m) and dimensions (w_m, h_m). These spatial relationships, combined with the interconnection information between modules, directly influence module mismatches on threshold voltage $V_{T0,i}$ and transconductance parameters β_i. We leverage these features to model how mismatches impact the final post-layout performance.

Point-to-Module Connections: If a module $v_{M_k} \in \mathcal{V}_M$ connects pin access points $v_{AP_1}, v_{AP_2}, \ldots, v_{AP_l} \in \mathcal{V}_{AP}$, we add the edges $(v_{AP_1}, v_{M_k}), (v_{AP_2}, v_{M_k}), \ldots, (v_{AP_l}, v_{M_k})$ to $\mathcal{E}_{PM}$. More importantly, $\mathcal{E}_{PM}$ bridges the gap between point-to-point and module-to-module message passing, fusing physical and logical information. According to (3), layout affects performance through two key aspects: 1) interconnect and 2) mismatch parasitic effects. These aspects are inherently intertwined with nonlinear effects. Thus, our performance estimation model must consider how these features interact with each other. To capture these nonlinear effects, we incorporated the connection relationships between pin access points and modules into our model.

C. 3DGNN-Based Classifier Network Architecture

Analog routing modeling encounters challenges with discreteness, sparsity, and low resolution compared to placement. The 2-D-based routing guidance map in GeniusRoute falls short of offering effective solutions. By leveraging insights from protein structure prediction, we seek to employ advanced techniques [24] in generating analog routing guidance.

To tackle those challenges above, we present a novel and protein-inspired 3DGNN framework that effectively incorporates the distance relationships between various pin access points, including their connectivity to modules. A key factor that requires particular attention is the routing resource competition among pin access points from different nets. In the expert's routing design experience, the distance between the pin access points and the modules will also be considered [8]. Thus, we use the distance-augmented module to augment edges from pin access points to modules, and from pin access points to pin access points.

Distance Augmented Module: In 3DGNN, the most important component is the Distance Augment module, where we embed the cost-aware distances between different nodes into the messages between the existing nodes.

The embedding of the pin access point/module v_{AP_k}/v_{M_k} is obtained by aggregating each incoming message e_k. The message e_k is updated based on $\mathcal{E}_{sk}$, the set of incoming messages pointing to the pin access point/module v_{AP_k}/v_{M_k}. The distance between v_k and v_s can be naturally decomposed into the horizontal distance h, the vertical distance w, and the distance in the z-axis direction z. Furthermore, we can naturally incorporate the routing cost in each direction into the calculation of the distance formula, thereby obtaining a cost-related distance function to measure the routing cost distance between different pin access points as shown in Fig. 7(a). Specifically, we can define the distance honoring routing cost as follows:

$$d_{\text{cost}}(v_k, v_s) = \sqrt{(C_k[0]\cdot h_{ks})^2 + (C_k[1]\cdot w_{ks})^2 + (C_k[2]\cdot z_{ks})^2} \quad (9)$$

where C_k is the routing guidance assigned for pin access point v_k, $h_{ks}/w_{ks}/z_{ks}$ is the distance between v_k and v_s along horizontal/vertical/Z-axis direction.

Without further processing, directly embedding the distances onto the original messages may lead to a plateau at the beginning of training as the initial network tends to behave linearly. We avoid this by expanding the distance with radial basis functions as in [24]. With the definition of cost-aware distance, we can define the aggregation function used to encode 3-D position information as in (10), which converts the position information r_h to an embedding of distance with radial basis functions

$$\rho^{p\rightarrow e}\left(\{r_h\}_{h=v_k\cup v_k}\right) = \Psi(d_{\text{cost}}(v_k, v_s)) = \exp\left(-\gamma\,||d_{\text{cost}}(v_k, v_s) - \mu_k||^2\right). \quad (10)$$

Cost-Aware Message Passing: In 3DGNN, the proposed cost-aware message passing is shown in Fig. 7(c) and Algorithm 1, which can be defined as

$$e_k^l = \phi^e\left(e_k, v_{r_k}, v_{s_k}, \mathcal{E}_{s_k}, \rho^{p\rightarrow e}\left(\{r_h\}_{h=r_k\cup s_k}\right)\right)$$
$$v_i^l = \phi^v\left(v_i, \rho^{e\rightarrow v}\left(\mathcal{E}_i^l\right)\right), u^l = \phi^u\left(u, \rho^{v\rightarrow u}\left(\mathcal{V}^l\right)\right) \quad (11)$$

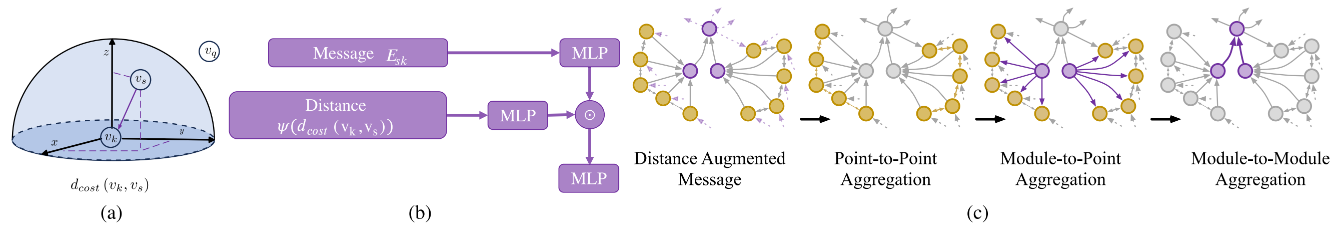

v_q
v_s
v_k
$d_{cost}(v_k, v_s)$
(a)
Message E_{sk}
Distance $\psi(d_{cost}(v_k, v_s))$
MLP
MLP
MLP
(b)
Distance Augmented Message
Point-to-Point Aggregation
Module-to-Point Aggregation
Module-to-Module Aggregation
(c)

Fig. 7. Important components of our proposed 3DGNN model. (a) Routing cost distance. (b) Distance augmented module. (c) Cost-aware message passing procedure on the heterogeneous graph.

where ϕ^e, ϕ^v, and ϕ^u are three information update functions on edges, pin access points/modules, and the whole graph, respectively. $\rho^{p\rightarrow e}$ and $\rho^{v\rightarrow u}$ aggregate information between different types of geometries. Especially, the 3-D information in $\boldsymbol{r}_h$ is converted and incorporated to update each cost-aware message e_k.

In addition to the distance augmented module $\rho^{p\rightarrow e}$, the message transformation function ϕ^e used is

$$\mathrm{MLP}\big(\mathrm{MLP}\big(v_{r_k}\big) \odot \mathrm{MLP}(\Psi(d_{\text{cost}}(v_k, v_s)))\big) \tag{12}$$

where MLP denotes a Multilayer Perception layer with weights and bias and $\odot$ denotes the element-wise multiplication.

The aggregation function $\rho^{e\rightarrow v}$ is defined as the sumation of received message $\rho^{e\rightarrow v}(\mathcal{E}_i^l) = \sum_{(\boldsymbol{e}_k^l, r_k, s_k)\in\mathcal{E}_i^l} \boldsymbol{e}_k^l$. The combination function ϕ^v is defined using the sum as $v_i + \sum_{(\boldsymbol{e}_k^l, r_k, s_k)\in\mathcal{E}_i^l} \boldsymbol{e}_k^l$. The global feature $\boldsymbol{u}$ is updated based on the final node features V^T and the function is $\phi^u = \sum_{i=1}^n \mathrm{MLP}(v_i^T)$.

Training Objectives: After L layers of cost-aware message passing, we add a fully connected layer FC to enable the network to predict the performance metrics corresponding to different routing guidance and placements. The overall prediction objective can be defined as follows:

$$\hat{\boldsymbol{y}}_{\text{OV}}, \hat{\boldsymbol{y}}_{\text{CMRR}}, \hat{\boldsymbol{y}}_{\text{UGB}}, \hat{\boldsymbol{y}}_{\text{Gain}}, \hat{\boldsymbol{y}}_{\text{Noise}} = f_{\boldsymbol{\theta}_C}(\mathcal{G}_H, \boldsymbol{C}) \tag{13}$$

where $\hat{\boldsymbol{y}}_{\text{OV}}$, $\hat{\boldsymbol{y}}_{\text{CMRR}}$, $\hat{\boldsymbol{y}}_{\text{UGB}}$, $\hat{\boldsymbol{y}}_{\text{Gain}}$, and $\hat{\boldsymbol{y}}_{\text{Noise}}$ represent the predicted metrics of OV, common-mode rejection ratio, unity-gain bandwidth, gain, and noise, respectively. We utilize the L_2 Loss as the loss function for the model.

D. Routing Performance Potential Modeling and Relaxation

To realize routing guidance that minimizes the constructed potential, we created a differentiable model to predict the post-layout performance of the routing guidance in the target design. The complete potential to be minimized is the sum of the FoM results of the predicted performance and constraint penalty.

FoM: We use the FoM to create a differentiable performance objective function that balances various performance targets, influencing routing guidance in post-layout simulations. Formally, the potential function associated with the post-layout performance related to routing guidance can be defined as follows:

$$V(\boldsymbol{C}) = \boldsymbol{w}_{\text{FoM}} \cdot f_{\boldsymbol{\theta}}\left(\mathcal{G}_H^{\text{val}}, \boldsymbol{C}\right) + g(\boldsymbol{C}) \tag{14}$$

where $\boldsymbol{w}_{\text{FoM}}$ represents the weight preferences for different metrics corresponding to a given FoM, and $g(\boldsymbol{C})$ describes the constraint function related to the routing guidance.

There are many relevant discussions regarding selecting the FoM in analog design [29]. FoMs can be categorized into uniform and nonuniform weights. Nonuniform weights allow for flexible prioritization of design objectives like OV and gain, enabling more realistic modeling but complicating the optimization process. The complexity of optimizing the performance model often leads to sacrifices in other metrics. Thus, the nonuniform weight requires significant expertise or iterative tuning to determine effective values. For simplicity, we applied uniform FoM in our main experiments. We also conducted ablation studies to evaluate the impact of FoM selection on final post-layout performance in Section V-E.

Potential Modeling: For the constraint function $g(\boldsymbol{C}_i)$, we have employed an interior point penalty function approach, which can be defined as follows:

$$g(\boldsymbol{C}_i) = -r\sum_{j=1}^{3}(\log \boldsymbol{C}_i[j] + \log(c_{\max} - \boldsymbol{C}_i[j])) \tag{15}$$

where r is a small positive value that ensures when x approaches the boundary of the feasible region D, $g(\boldsymbol{C}_i) \rightarrow +\infty$, and when x is within D, $g(\boldsymbol{C}_i) \approx 0$. The feasible region D of (15) implies that the elements in the routing guidance must be greater than or equal to 0 and less than or equal to $c_{\max}$.

Potential Relaxation: As every term in $V(\cdot)$ is differentiable concerning the routing guidance, given an initial set of routing guidance $\boldsymbol{C}_{\text{init}}$ sampled from a given distribution, we can minimize $V(\boldsymbol{C})$ using a gradient descent algorithm, such as L-BFGS. We repeat the optimization multiple times with different initializations. A pool of the N_{pool} lowest-potential structures is maintained and once full, we initialize $p_{\text{relax}} \cdot N_{\text{pool}}$ of routing guidance from those with noise added to the routing guidance. Finally, we will derive the top-N_{derive} routing guidance results with optimal potential values.

E. 3DGRU-Based Generator Network Architecture

Our approach extends the variational graph autoencoder [30] to analog routing by introducing a suitable encoder. Deviating from previous work GeniusRoute [15] and PARoute [21], we interpret the integral routing guidance prediction as a link prediction task being built from paths chosen from pin access points of each net.

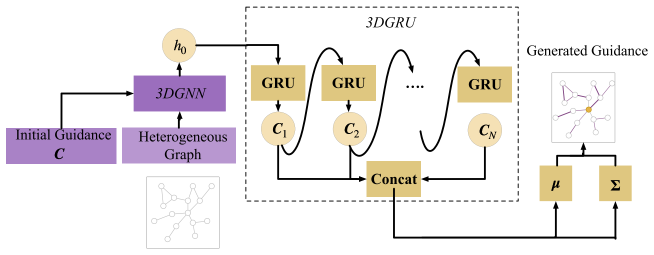


Fig. 8. Structure of 3DGRU-based routing guidance generator.

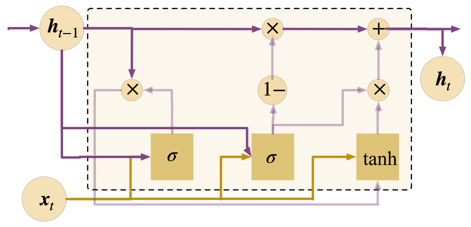


Fig. 9. Structure of gated recurrent unit.

The key advantage of this view is that the decoder can create the routing guidance by sequentially honoring a given routing order, rather than trying to build the routing guidance by pin access points at once neglecting their construction dependence. To connect all the nets considering design rules and constraints, the existing router used to adopt the negotiation-based rip-up and reroute methodology to alleviate congestion as in [20].

We follow the net order defined in the existing router, which is defined as follows:

$$R_i = \alpha \cdot \text{HPWL}_i + \beta \cdot |P_i| + \gamma \cdot d_i \tag{16}$$

where R_i represents the ranking score of a net n_i in the queue. HPWL_i is the half perimeter wire length of n_i according to its pins, d_i is the estimated degree of symmetry if n_i is a symmetry (resp. self-symmetry) net and α, β, γ are weighted constants.

Honoring the routing order, we encode $\boldsymbol{C}$ with a graph recurrent message-passing network, similar to [31]. The generated routing guidance vector $\boldsymbol{m}_{ij}$ for each pin access point ap_i is updated as

$$\boldsymbol{C}_i = \text{GRU}\left(\boldsymbol{x}_i, \{\boldsymbol{m}_{ki}\}_{k\in N(i)\backslash j}\right) \tag{17}$$

where GRU is a Gated Recurrent Unit [32] adapted for routing guidance message passing as show in Fig. 9.

The computational process is processed in a sequential way to capture the independence between different routing paths, which is expressed as follows:

$$\begin{aligned}
\boldsymbol{s}_{ij} &= \sum_{k\in N(i)\backslash j} \boldsymbol{m}_{ki}, \boldsymbol{z}_{ij} = \sigma\left(\boldsymbol{W}^z\boldsymbol{x}_i + \boldsymbol{U}^z\boldsymbol{s}_{ij} + \boldsymbol{b}^z\right)\\
\boldsymbol{r}_{ki} &= \sigma\left(\boldsymbol{W}^r\boldsymbol{x}_i + \boldsymbol{U}^r\boldsymbol{m}_{ki} + \boldsymbol{b}^r\right)\\
\boldsymbol{m}_{ij} &= \left(1 - \boldsymbol{z}_{ij}\right) \odot \boldsymbol{s}_{ij} + \boldsymbol{z}_{ij} \odot \widetilde{\boldsymbol{m}}_{ij}\\
\widetilde{\boldsymbol{m}}_{ij} &= \tanh\left(\boldsymbol{W}\boldsymbol{x}_i + \boldsymbol{U}\sum_{k\in N(i)\backslash i} \boldsymbol{r}_{ki} \odot \boldsymbol{m}_{ki}\right)
\end{aligned} \tag{18}$$

where $\boldsymbol{W}^z$, $\boldsymbol{W}^r$, $\boldsymbol{W}$ represent the weight of the forget, input, and output part of the GRU unit, respectively.

Message passing adheres to a schedule where $\boldsymbol{m}_{ij}$ is computed only after all its precursor values $\{\boldsymbol{m}_{ki} \mid k \in N(i)\backslash j\}$ have been calculated. The design of the 3DGRU structure incorporates the generation order of the routing order in (16), allowing the sequence dependencies to be considered when generating routing guidance.

F. Performance-Driven Routing Guidance Generation

We introduce a variational posterior distribution $q(\boldsymbol{\theta}_G|\mathcal{G}_H)$ that conditions solely on the initial routing guidance distribution, making it computationally efficient for simulating the prior distribution. From this, we can derive the evidence lower bound (ELBO) of our objective function as follows:

$$\begin{aligned}
\log p(y|\mathcal{G}_H, \mathcal{N}) &= \log \mathbb{E}_q\left[p(y|\mathcal{G}_H, \boldsymbol{\theta}_G)\frac{p(\boldsymbol{\theta}_G|\mathcal{N})}{q(\boldsymbol{\theta} G|\mathcal{G}_H)}\right]\\
&\geq \mathbb{E}_q\left[\log p(y|\mathcal{G}_H, \boldsymbol{\theta}_G)\right] - \text{KL}(q(\boldsymbol{\theta}_G|\mathcal{G}_H)\|p(\boldsymbol{\theta}_G|\mathcal{N}))).
\end{aligned} \tag{19}$$

The generative objective can then be formulated as

$$\begin{aligned}
L_{\text{gen}}(\boldsymbol{C}|\mathcal{G}_H, \mathcal{N}) &= \mathbb{E}\left[\log p(\boldsymbol{C}|\boldsymbol{\theta}_G)\right]\\
&\quad - \text{KL}(p(\boldsymbol{\theta}_G|\boldsymbol{C})||q(\boldsymbol{\theta}_G|\mathcal{N}))
\end{aligned} \tag{20}$$

where $\text{KL}(p(\boldsymbol{\theta}_G|\boldsymbol{C})||q(\boldsymbol{\theta}_G|\mathcal{N}))$ computes the Kullback–Leibler divergence between the distributions of the original and generated routing guidance.

Next, we present the construction of p and q for optimization. Following established practices in variational inference [33], we model the variational posterior distribution as a Gaussian

$$\boldsymbol{W}_\theta \sim N(\mu(\boldsymbol{u}), \Sigma(\boldsymbol{u})) \tag{21}$$

where $\boldsymbol{W}_q$ is the linear layer generated by distribution q, $\boldsymbol{u}$ are the vector representation for the heterogeneous graph $\mathcal{G}'_H$ and routing guidance $\boldsymbol{C}$. $\mu(\cdot)$ and $\Sigma(\cdot)$ are two MLPs. In essence, we utilize these two learnable networks to output the mean $\mu(\cdot)$ and variance $\Sigma(\cdot)$ of the distribution of W_q, as illustrated in Fig. 8.

To compute the first term in (20), we employ Monte Carlo sampling to draw K samples from $q(\boldsymbol{W} \mid \mathcal{G}'_H)$, denoted as $\{\boldsymbol{W}^i_q\}^K_{i=1}$. For each sampled $\boldsymbol{W}^i_q$, we derive a routing guidance prediction $\hat{\boldsymbol{C}}_i$. The ELBO objective can thus be expressed as

$$\begin{aligned}
L_{\text{ELBO}}\left(\boldsymbol{C}, \mathcal{G}'_H, \mathcal{N}\right) &= L^d\left(\boldsymbol{C}, \hat{\boldsymbol{C}}'\right) - \text{KL}\left(\boldsymbol{C}, \hat{\boldsymbol{C}}'\right)\\
&= \frac{\eta}{K}\sum_{i=1}^{K} L^d\left(\boldsymbol{C}, \hat{\boldsymbol{C}}_i\right) - \text{KL}\left(q\left(\boldsymbol{W} \mid \mathcal{G}'_H\right)\|p(\boldsymbol{W} \mid \mathcal{N})\right)
\end{aligned} \tag{22}$$

where L^d is a function that measures distance. For continuous values, we typically employ the L2 distance, while for discrete values, the cross-entropy function may be more appropriate.

For performance estimation, we define the loss function as

$$L_{\text{cls}}(\boldsymbol{C}) = \boldsymbol{w}_{\text{FoM}} \cdot f_{\boldsymbol{\theta}_C}(\mathcal{G}_H, \boldsymbol{C}) \tag{23}$$

where $\boldsymbol{w}_{\text{FoM}}$ indicates the weight preferences for different metrics related to a specific specification.

TABLE I
COMPARISON OF DIFFERENT ROUTING GUIDANCE GENERATION METHODS

Method	Guidance Format	3D Information	Generator	Discriminator	Optimizer	Optimizee	Repeated Opt
GeniusRoute [15]	Uniform	×	✓	×	×	×	×
PARoute [21]	Non-uniform	✓	×	✓	L-BFGS	Individual Guidance	✓
PARoute2	Non-uniform	✓	✓	✓	SGD	Guidance Generator	✓

TABLE II
BENCHMARK CIRCUITS INFORMATION

Metrics	OV ↓ (μV)	CMRR ↑ (dB)	BW ↑ (MHz)	Gain ↑ (dB)	Noise ↓ (μV_{rms})
OTA1	-	155.30	108.10	37.19	213.40
OTA2	-	37.72	87.54	34.26	213.60
OTA3	-	126.60	590.00	47.50	3061.00
OTA4	-	110.50	802.10	44.20	3430.00
Metrics	Frequency↑(MHz)			Power↓(μW)	
OSC	17.36			403.5	

In each iteration, we train the routing guidance generation model using L_{gen}, subsequently selecting the generated routing guidance with L_{cls} and updating the generator accordingly. Through (20), we progress toward $\max_{\theta_P} L_{\text{gen}}(f_M(\boldsymbol{C} \mid \boldsymbol{\theta}_M), \boldsymbol{C})$ in (8). In turn, (23) directs routing generation to minimize the objectives outlined in (8).

G. Comparison With Previous Methods

Table I highlights the differences between PARoute2 and existing models for routing guidance generation. GeniusRoute [15] adopts a uniform-grid-based guidance format and uses a generator to produce guidance. Since the generator lacks performance-related estimation capabilities, the routing guidance generated by GeniusRoute cannot directly lead to performance improvements. In contrast, PARoute [21] improves upon this by using nonuniform routing guidance and employs a performance discriminator for optimization. However, PARoute [21] also has a significant limitation because it relies solely on a discriminator, requiring a complex gradient-based L-BFGS method for optimizing each routing guidance. This necessitates multiple optimizations for each routing guidance, involving derivative calculations that demand substantial computational resources and time.

In contrast, PARoute2 employs an alternating optimization approach that integrates a generative model with a performance discriminator to produce routing guidance. The generative model quickly generates candidate guidance, while the performance discriminator evaluates and scores them. Based on the scoring results, the generative model adjusts it generation probability of high-scoring guidances via (22). This optimization optimizes the generator as a whole instead of optimizing each routing guidance individually.

V. EXPERIMENTS

A. Experimental Setting

We implement the proposed performance-driven analog placement framework with Python 3.7.5 and C++. The main part of the analog placement and routing algorithms are implemented in C++ based on the MAGICAL framework [34]. The 3DGNN training and cost guide relaxation are implemented with torch-2.0.0 [35]. All designs are in TSMC 40nm technology, extracted for parasitics with Calibre PEX, and simulated with Cadence Spectre.

Benchmarks: The details of the experimental benchmarks used in this paper are described in Table II. We also add an Oscillator (OSC) design to investigate the transferability of our framework. We conducted experiments on four OTAs benchmarks OTA1, OTA2, OTA3, and OTA4. OTA1 and OTA2 are 2-stage miller-compensated amplifier OTAs, as shown in Fig. 10(a). OTA3 and OTA4 are two 2-stage telescopic OTA, and have a more complex topology, as depicted in Fig. 10(b).

Net Weights: As mentioned in [36], different net weighting combinations could significantly change the floor plan and placement solutions. We thus adjust the net weights to generate different placement results as in Table IV. Following [17], we classify the modules within a net based on their functionality as follows.

1) *Functional Modules:* first/other stage devices;
2) *Common Feedback and Bias Modules:* common-mode feedback (CMFB) devices, and bias devices;
3) *Compensation Modules:* passive devices in the compensation feedback loop, such as miller capacitance.

Based on the classification, our net weights settings consist of the following variations: A (uniform net weights for all modules), B (increased net weights for the functional modules), and C (increased net weights for the compensation modules). Due to its characteristics, the Miller-compensated OTAs include special compensation feedback devices in the compensation feedback loop. The Telescopic OTAs (OTA3 and OTA4) do not contain compensation circuits, thus they only include A/B settings.

Compared Methods: In our experiments, we compare our PARoute framework with the following strong baselines: 1) *MagicalRoute:* a default router of the MAGICAL [34] framework proposed in [20] and 2) *GeniusRoute:* [15]. MagicalRoute [20] incorporates industrial design rules and analog-specific geometric and electrical constraints to enhance the overall circuit performance. GeniusRoute [15] is the recent work on using ML models to guide the analog routing process with generation model VAE, which is the closest one to our work.

Evaluation Metrics: The proposed approaches aim to boost the ability to obtain analog circuit layouts with optimal performance. The well-known post-layout metrics **OV**, **CMRR**, DC Gain (**Gain**), Bandwidth (**BW**), and Noise and the runtime (i.e., wall-clock time) are used to measure performance and efficiency, respectively. We use 5 hosts to collect data at the same time. We use 2000 samples on target design with different placements and routing solutions to train PARoute.

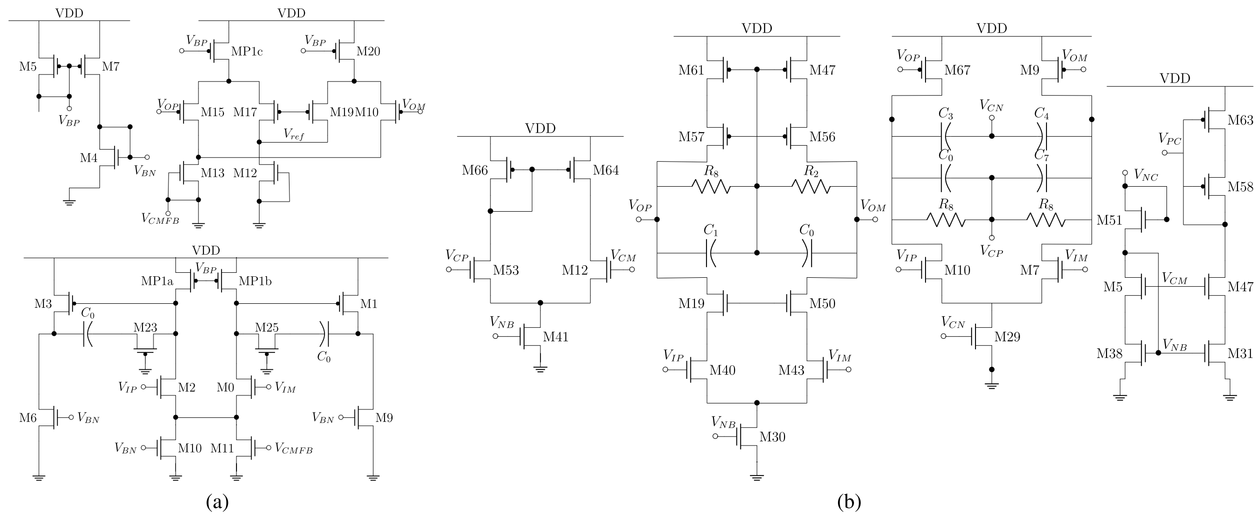


Fig. 10. Topology description of two designs with two sizings. (a) Miller-compensated operational transconductance amplifiers (OTA1 and OTA2). (b) Telescopic operational transconductance amplifiers (OTA3 and OTA4).

TABLE III
BENCHMARK CIRCUITS INFORMATION

Benchmark	#PMOS	#NMOS	#Cap	#Res	#Total
OTA1	6	8	2	0	25
OTA2	6	8	2	0	25
OTA3	16	10	6	4	36
OTA4	16	10	6	4	36
OSC	3	4	4	3	14

TABLE IV
NET WEIGHTS SETTING FOR DIFFERENT BENCHMARK

Bench	Net Weights
OTA1-A	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
OTA1-B	[1, 1, 1, 1, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 2, 1, 2, 2]
OTA1-C	[1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
OTA2-A	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
OTA2-B	[1, 1, 1, 1, 2, 2, 1, 1, 2, 2, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 2, 1, 2, 2]
OTA2-C	[1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
OTA3-A	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,1, 1, 1, 1, 1, 1]
OTA3-B	[2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 2, 2,2, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,2, 2, 1, 1, 1, 1]
OTA4-A	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,1, 1, 1, 1, 1, 1]
OTA4-B	[2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 2, 2,2, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,2, 2, 1, 1, 1, 1]

* OTA3 and OTA4 only include A/B settings, as they do not contain the compensation modules.

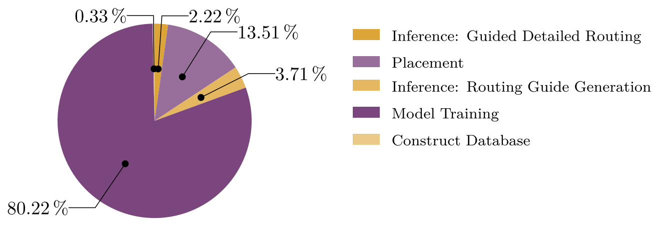
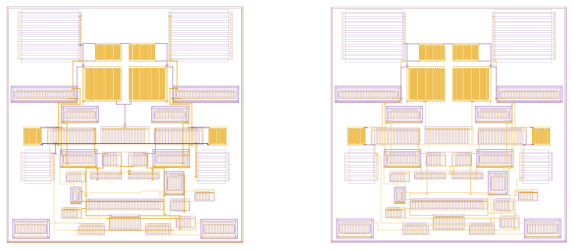


Fig. 11. Runtime breakdown of PARoute for OTA1.



(a) (b)

Fig. 12. GeniusRoute and PARoute routing solutions. (a) GeniusRoute routing solution; (b) PARoute routing solution.

B. Routing Performance

We use the default analog placer in the MAGICAL framework [34] to generate different placement results for each design. Then, we compare the performance of routing solutions generated by our proposed method with other baseline methods. To verify the results on circuit performance, we conduct post-layout simulations. Calibre PEX is used to extract the parasitic resistance, parasitic capacitor, and coupling capacitance (R+C+CC). Then, the performance is evaluated with Cadence Spectre.

Results in Table V demonstrate that our proposed PARoute and PARoure-V2 method obtains routing solutions that are superior to baseline methods in terms of post-layout performance. As shown in Table V, PARoute achieves up to 3671.00 μV, 30.33 dB, 169.2 MHz, 38.141 dB, and 2028 μV_{rms} improvement compared to the GeniusRoute [15] concerning OV, CMRR, BandWidth, DC Gain, and Noise.

PARoute exhibits enhanced stability when considering the potential post-layout performance. This is especially evident in the corner case scenario of OTA4-A, where both the optimization-based MagicalRoute [20] and the experience-based GeniusRoute [15] failed to improve the overall post-layout performance. In contrast, our proposed method has successfully identified the possibility of further improvements, achieving an enhanced bandwidth of 169.2 MHz and a gain of 38.14 dB.

TABLE V
COMPARISONS BETWEEN BASELINE METHODS AND THE PROPOSED METHOD

Bench	GeniusRoute [15]					PARoute [21]					PARoute2				
	OV↓	CMRR ↑	BW ↑	Gain ↑	Noise ↓	OV↓	CMRR ↑	BW ↑	Gain ↑	Noise ↓	OV↓	CMRR ↑	BW ↑	Gain ↑	Noise ↓
OTA1-A	1464	89.01	50.97	36.79	2254	1319	112.9	51.02	36.83	226.0	**1303**	**121.6**	**51.12**	**36.87**	**225.0**
OTA1-B	1121	85.97	48.64	36.59	2279	1031	116.3	48.65	36.62	228.0	**727.2**	**134.7**	**50.00**	**36.94**	**226.0**
OTA1-C	70.61	93.56	49.87	36.91	2253	47.47	101.9	49.86	**36.92**	2254	**15.91**	**117.2**	**49.88**	36.91	**2253**
OTA2-A	7032	13.94	34.36	14.6	439.6	3361	22.06	**35.68**	21.71	303.3	**3135**	**22.50**	35.58	**22.15**	**299.40**
OTA2-B	3562	20.97	35.4	20.87	313.8	**3135**	**22.5**	**35.58**	**22.15**	**299.4**	3446	21.3	35.48	21.17	310.50
OTA2-C	7590	39.79	37.8	33.53	264.7	7558	40.1	38.03	33.63	265.4	**7539**	**40.2**	**38.03**	**33.65**	**265.0**
OTA3-A	29.83	93.35	267.6	6.066	2295	**0.249**	110.7	396.9	47.66	2278	0.250	**110.9**	**397.2**	**47.66**	**2259**
OTA3-B	230.8	46.94	51.42	6.86	1685	210.6	46.97	**51.75**	6.949	1693	**207.8**	**46.97**	48.98	6.850	**1558**
OTA4-A	24.14	139.1	294.8	6.079	2448	**0.003**	116.3	464.0	44.22	2343	0.240	**121.60**	**469.1**	**45.28**	**2381**
OTA4-B	11.17	93.34	303.5	6.078	2551	1.137	123.5	304	6.078	2557	**0.460**	**127.9**	**468.3**	**42.97**	**2391**
Average	1967.16	62.70	112.34	16.76	1452.91	1534.45	70.03	142.45	25.59	1222.11	**1507.19**	**74.33**	**159.26**	**29.36**	**1194.29**
Ratio	1.000	1.000	1.000	1.000	1.000	0.780	1.117	1.268	1.527	0.841	**0.766**	**1.186**	**1.418**	**1.752**	**0.822**

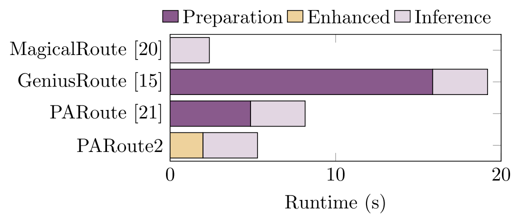


Fig. 13. Runtime comparison between different methods.

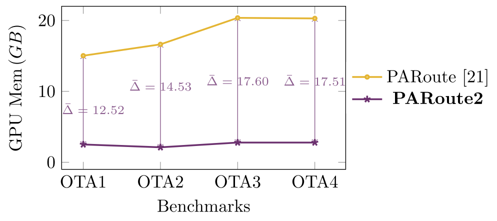


Fig. 14. Average peak GPU memory usage comparison across different benchmarks.

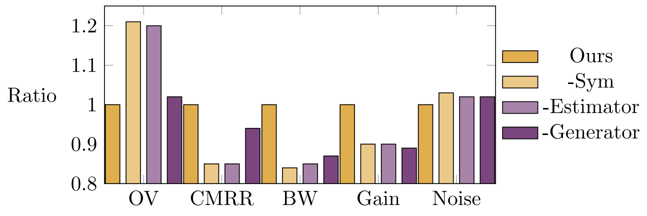


Fig. 15. Comparison among different modules of the averaged result across all benchmarks, including OV, CMRR, Bandwidth, DC gain, and Noise.

For the PARoute2 framework, the Performance-Driven Routing Guidance generation strategy allows for higher-quality Routing Guidance with lower inference times. Specifically, compared to GeniusRoute, PARoute2 achieves improvements of up to 3897.00 μV, 48.73 dB, 174.3 MHz, 39.20 dB, and 2053 μV_{rms} in terms of OV, CMRR, BandWidth, DC gain, and Noise, respectively. On average, PARoute2 achieves a 23.4% reduction in OV, an 18.6% increase in CMRR, a 41.8% increase in BandWidth, a 75.2% increase in DC Gain, and a 17.8% reduction in Noise compared to GeniusRoute. In comparison to PARoute, PARoute2 shows average improvements of 1.39%, 6.85%, 14.96%, 22.46%, and 1.91% in OV, CMRR, BandWidth, DC Gain, and Noise, respectively.

C. Comparison With the MAGICAL Router

We not only compare our method with the learning-based rivals but also perform a performance comparison of our results with the post-simulation results of the default Magical Router (MagicalRoute [20]). We analyzed the performance gains of the Learning-based method compared to MagicalRouter, visualized in Fig. 16. The performance comparison reveals that learning-based methods significantly enhanced the post-simulation performance of MagicalRoute. While experience-based GeniusRoute achieved a notable increase in CMRR, it failed to deliver average improvements in Bandwidth and Noise metrics. In contrast, PARoute and PARoute2, guided by post-simulation performance, demonstrated substantial enhancements across all four metrics.

D. Runtime and Computing Resource Comparison

Fig. 13 plots the runtime breakdown of our proposed method on OTA1 designs. The most consuming part is the model training part, which takes 80.22% of the total runtime. Furthermore, it takes 3.71% of the total time for the routing cost generation, which includes feature extraction, inference, and potential relaxation. Even though the average runtime of our proposed approach is 7.48× slower than MagicalRoute [20], it is nearly 2.29× faster than GeniusRoute [37] due to the simplified 3-D graph structure.

As for PARoute2, we have replaced the time-consuming performance relaxation process with a performance-driven routing guidance generation process. PARoute2 has achieved better runtime results than PARoute, as shown in Fig. 13. The model generation speed in PARoute2 is 2.453× faster than the original Relaxation process on average. This results in an overall inference time where PARoute2's inference speed is nearly 1.545× faster than PARoute. We also compare the average peak GPU memory corresponding to two different routing guidance generation methods, as shown in Fig. 14. We measured the peak GPU memory usage and then calculated the average result for each design. The results indicate that

TABLE VI
COMPARISONS BETWEEN DIFFERENT TYPES OF FoM

Bench	Unifrom FoM [-1, 1, 1, 1, -1]					Non-uniform FoM-1 [-3, -3, 3, 1, -1]					Non-uniform FoM-2 [-1, -1, 1, 3, -3]				
	OV↓	CMRR ↑	BW ↑	Gain ↑	Noise ↓	OV↓	CMRR ↑	BW ↑	Gain ↑	Noise ↓	OV↓	CMRR ↑	BW ↑	Gain ↑	Noise ↓
OTA1-A	1303.00	121.60	51.12	36.87	225.00	1291.00	109.90	51.03	36.88	225.10	1489.00	135.80	50.16	36.98	226.00
OTA1-B	727.20	134.70	50.00	36.94	226.00	675.60	88.47	51.26	37.07	224.40	831.42	135.80	50.16	36.98	226.00
OTA1-C	15.91	117.20	49.88	36.91	2253.00	4.76	107.70	49.88	36.92	225.40	20.46	114.30	49.85	36.92	225.40
OTA2-A	3135.00	22.50	35.58	22.15	299.40	2107.00	17.57	35.66	23.41	365.80	3217.00	30.92	36.49	29.02	270.10
OTA2-B	3446.00	21.30	35.48	21.17	310.50	2456.00	25.54	36.21	26.85	286.80	4191.00	21.85	34.74	21.62	306.20
OTA2-C	7539.00	40.20	38.03	33.65	265.00	6999.80	33.86	37.26	30.82	270.30	8531.00	40.23	38.03	33.67	265.30
OTA3-A	0.25	110.90	397.20	47.66	2259.00	0.19	114.80	397.80	47.50	2272.00	0.18	115.70	397.60	47.98	2289.00
OTA3-B	207.80	46.97	48.98	6.85	1558.00	190.29	38.31	27.14	6.07	2344.00	411.90	34.33	21.96	8.35	887.00
OTA4-A	0.24	121.60	469.10	45.28	2381.00	0.77	118.10	477.70	43.91	2396.00	0.52	131.30	479.00	43.99	2432.00
OTA4-B	0.46	127.90	468.30	42.97	2391.00	0.85	126.90	492.00	43.79	2462.00	-0.71	127.70	442.70	43.76	2478.00
Average	1507.19	74.33	159.26	29.36	1194.29	1243.53	67.13	160.49	29.63	1084.67	1720.28	75.21	155.05	30.23	937.90
Ratio	1.00	1.00	1.00	1.00	1.00	0.83	0.90	1.01	1.01	0.91	1.14	1.01	0.97	1.03	0.79

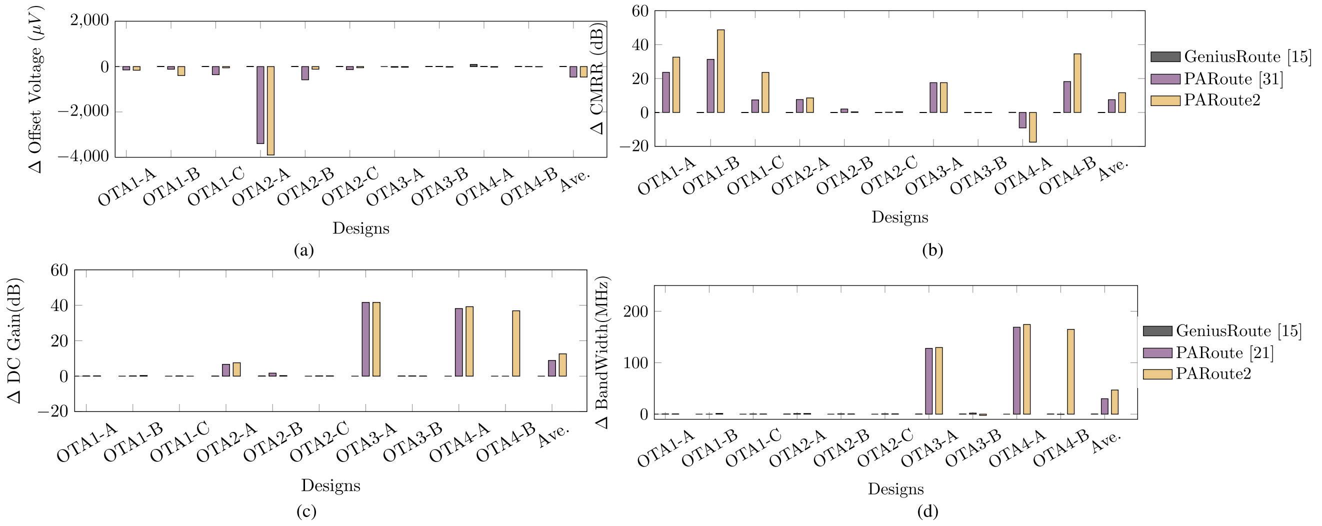


Fig. 16. Post-layout performance improvement compared with Magical Router [20]. (a) OV. (b) CMRR. (c) Bandwidth. (d) DC gain.

PARoute2 offers substantial GPU memory savings across all benchmarks compared to the previous solution, achieving an average of 15.54 GB GPU memory reduction.

E. Ablation Study

Effectiveness of Different Modules: To verify the effectiveness of the symmetry constraints (-Sym), the GNN-based performance estimator (-Estimator), and the GRU-based routing guidance generator (-Generator), we conduct ablation studies as shown in Fig. 15. We can observe that, without any of them, the obtained performance drops by a substantial margin, demonstrating the effectiveness of all of them. Additionally, we can observe that the Symmetry and Estimator contribute the most to the overall performance. It is worth mentioning that the Generator module also has a significant impact on the final performance results.

The Selection of Different FoMs: To further investigate the impact of different forms of FoM on the final performance, we compared the effects of different FoM types on the outcomes, including a uniform FoM and two nonuniform FoMs. As shown in Table VI, although the nonuniform FoM improves the specified preferred metrics, this enhancement often results in a decline in one or more other metrics. From the experimental results, this tradeoff is often complex and nonlinear. This implies that expert experience and effective FoM adjustments can further enhance the final post-simulation performance.

F. Discussion About Limitation on Transferability

In this section, we explore the limitations of the proposed performance-driven framework in terms of its transferability. Specifically, the varying performance metrics across different circuits pose significant challenges to the performance model's predictions. As shown in Table VII, we experimented with different transfer settings: varying sizing, topologies, and designs. For the different design settings, we extracted the weights of the pretrained model of the OTA1 design and fine-tuned the last layer of the model on the collected OSC data with 20 samples.

Under varying sizing and topology, the performance model often provides accurate predictions. However, when considering different designs, the performance model's predictions tend to exhibit substantial deviations. This limitation makes it difficult for our current framework to be directly transferred to other designs. A performance model capable of cross-design transferability would further enhance the framework's transferability.

TABLE VII
COMPARISONS BETWEEN DIFFERENT TRANSFER SETTINGS FOR PERFORMANCE MODEL PREDICTION.
(THE MSE IS MEASURED USING NORMALIZED RESULTS)

Transfer Setting	Design	Metrics	MSE↓
Different Sizing	OTA1 → OTA2	OV	0.6645
		CMRR	1.1766
		BW	0.0157
		Gain	0.0004
		Noise	0.0001
Different Topology	OTA1 → OTA3	OV	0.6157
		CMRR	1.3797
		BW	0.0335
		Gain	0.0484
		Noise	0.0182
Different Design	OTA1 → OSC	Frequency	3.32E+20
		Power	9.90E+02

VI. CONCLUSION

In this article, we present a new paradigm in analog IC routing, which learns routing guidance from solutions by an automatic routing engine. Our approach, PARoute2, combines nonuniform routing guidance generation and guided analog detailed routing. To address the key issue of generating effective guidance, we propose to model the performance potential and derive optimized routing guides. We further propose to perform routing guidance generation tasks using minimal GPU resources while achieving faster speeds. Experimental results show that our framework significantly improved multiple post-layout simulation metrics. Several promising directions can be considered. One important direction is to migrate the performance models across different types of analog circuits or technology nodes, thus we can improve the generalizability of the performance-driven routing to a wider range of circuit types.

REFERENCES

[1] M. P.-H. Lin, Y.-W. Chang, and C.-M. Hung, "Recent research development and new challenges in analog layout synthesis," in *Proc. ASPDAC*, 2016, pp. 617–622.

[2] H. Chen, M. Liu, X. Tang, K. Zhu, N. Sun, and D. Z. Pan, "Challenges and opportunities toward fully automated analog layout design," *J. Supercomput.*, vol. 41, no. 11, pp. 1674–4926, 1993.

[3] J. Crossley et al., "BAG: A designer-oriented integrated framework for the development of AMS circuit generators," in *Proc. ICCAD*, 2013, pp. 74–81.

[4] E. Chang et al., "BAG2: A process-portable framework for generator-based AMS circuit design," in *Proc. CICC*, 2018, pp. 1–8.

[5] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint generation for routing analog circuits," in *Proc. DAC*, 1991, pp. 561–566.

[6] E. Charbon, E. Malavasi, U. Choudhury, A. Casotto, and A. Sangiovanni-Vincentelli, "A constraint-driven placement methodology for analog integrated circuits," in *Proc. CICC*, 1992, pp. 28.2.1–28.2.4.

[7] H.-C. Ou, H.-C. C. Chien, and Y.-W. Chang, "Non-uniform multilevel analog routing with matching constraints," in *Proc. DAC*, 2012, pp. 549–554.

[8] L. Xiao, E. F. Young, X. He, and K. P. Pun, "Practical placement and routing techniques for analog circuit designs," in *Proc. ICCAD*, 2010, pp. 675–679.

[9] R. Martins, N. Lourenço, A. Canelas, and N. Horta, "Electromigration-aware and IR-drop avoidance routing in analog multiport terminal structures," in *Proc. DATE*, 2014, pp. 1–6.

[10] H.-Y. Chi, H.-Y. Tseng, C.-N. J. Liu, and H.-M. Chen, "Performance-preserved analog routing methodology via wire load reduction," in *Proc. ASPDAC*, 2018, pp. 482–487.

[11] J. M. Cohn, D. J. Garrod, R. A. Rutenbar, and L. R. Carley, "KOAN/ANAGRAM II: New tools for device-level analog placement and routing," *IEEE J. Solid-State Circuits*, vol. 26, no. 3, pp. 330–342, Mar. 1991.

[12] P.-H. Lin, H.-C. Yu, T.-H. Tsai, and S.-C. Lin, "A matching-based placement and routing system for analog design," in *Proc. VLSI-DAT*, 2007, pp. 1–4.

[13] H.-C. Ou, H.-C. C. Chien, and Y.-W. Chang, "Simultaneous analog placement and routing with current flow and current density considerations," in *Proc. DAC*, 2013, pp. 1–6.

[14] H. Chen et al., "AutoCraft: Layout automation for custom circuits in advanced FinFET technologies," in *Proc. ISPD*, 2022, pp. 175–183.

[15] K. Zhu et al., "GeniusRoute: A new analog routing paradigm using generative neural network guidance," in *Proc. ICCAD*, 2019, pp. 1–8.

[16] H. Chen et al., "Reinforcement learning guided detailed routing for custom circuits," in *Proc. ISPD*, 2023, pp. 26–34.

[17] M. Liu et al., "Towards decrypting the art of analog layout: Placement quality prediction via transfer learning," in *Proc. DATE*, 2020, pp. 496–501.

[18] A. W. Senior et al., "Improved protein structure prediction using potentials from deep learning," *Nature*, vol. 577, no. 7792, pp. 706–710, 2020.

[19] J. Jumper et al., "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[20] H. Chen, K. Zhu, M. Liu, X. Tang, N. Sun, and D. Z. Pan, "Toward silicon-proven detailed routing for analog and mixed-signal circuits," in *Proc. ICCAD*, 2020, pp. 1–8.

[21] P. Xu, G. Chen, K. Zhu, T. Chen, T.-Y. Ho, and B. Yu, "Performance-driven analog routing via heterogeneous 3DGNN and potential relaxation," in *Proc. 61st ACM/IEEE Design Autom. Conf.*, 2024, pp. 1–6.

[22] M. J. Pelgrom, A. C. Duinmaijer, and A. P. Welbers, "Matching properties of MOS transistors," *IEEE Journal Solid-State Circuits*, vol. 24, no. 5, pp. 1433–1439, Oct. 1989.

[23] K. Lampaert, G. Gielen, and W. M. Sansen, "A performance-driven placement tool for analog integrated circuits," *IEEE J. Solid-State Circuits*, vol. 30, no. 7, pp. 773–780, Jul. 1995.

[24] K. Schütt, P.-J. Kindermans, H. E. Sauceda, S. Chmiela, A. Tkatchenko, and K.-R. Müller, "SchNet: A continuous-filter convolutional neural network for modeling quantum interactions," in *Proc. 31st NIPS*, 2017, pp. 992–1002.

[25] J. Gasteiger, J. Groß, and S. Günnemann, "Directional message passing for molecular graphs," in *Proc. ICLR*, 2020, pp. 1–13.

[26] Y. Liu et al., "Spherical message passing for 3D molecular graphs," in *Proc. ICLR*, 2022, pp. 1–20.

[27] J. Edmonds, "Paths, trees, and flowers," *Can. J. Math.*, vol. 17, pp. 449–467, Jan. 1965.

[28] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint-based channel routing for analog and mixed analog/digital circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 12, no. 4, pp. 497–510, Apr. 1993.

[29] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Multi-objective Bayesian optimization for analog/RF circuit synthesis," in *Proc. DAC*, 2018, pp. 1–6.

[30] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *Proc. NIPS*, 2016, pp. 1–3.

[31] W. Jin, R. Barzilay, and T. Jaakkola, "Junction tree variational autoencoder for molecular graph generation," in *Proc. ICML*, 2018, pp. 2323–2332.

[32] K. Cho et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724–1734.

[33] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114*.

[34] H. Chen et al., "MAGICAL 1.0: An open-source fully-automated AMS layout synthesis framework verified with a 40-nm 1GS/s $\Delta\sum$ ADC," in *Proc. CICC*, 2021, pp. 1–2.

[35] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *Proc. NIPS*, vol. 32, 2019, pp. 8026–8037.

[36] M. Liu et al., "Closing the design loop: Bayesian optimization assisted hierarchical analog layout synthesis," in *Proc. DAC*, 2020, pp. 496–501.

[37] K. Zhu, H. Chen, M. Liu, X. Tang, N. Sun, and D. Z. Pan, "Effective analog/mixed-signal circuit placement considering system signal flow," in *Proc. ICCAD*, 2020, pp. 1–9.

Peng Xu received the B.S. degree from Central South University, Changsha, China, in 2019, and the M.S. degree from the Harbin Institute of Technology (Shenzhen), Shenzhen, China, in 2022. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, under the supervision of Prof. B. Yu.

His research interests include machine learning for analog physical design and optimization in EDA problems.

Jindong Tu received the B.S. degree from Shanghai University, Shanghai, China, in 2023. He is currently pursuing the Ph.D. degree with the School of Science and Engineering, The Chinese University of Hong Kong (Shenzhen), Shenzhen, China.

His research interests include analog circuit design automation and mix-signal IC design.

Guojin Chen received the B.Eng. degree in software engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2019. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

His current research interests include machine learning in VLSI design for manufacturability and physics-informed networks for solving EDA area problems.

Keren Zhu (Member, IEEE) received the B.S. degree in electrical engineering (with the highest distinction) from the University of Wisconsin–Madison, Madison, WI, USA, in 2016, and the Ph.D. degree from the Chandra Family Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA, in 2022.

He is currently an Associate Professor with the School of Microelectronics, Fudan University, Shanghai, China. His current research interests include physical design automation, analog integrated circuit design automation, and logic synthesis.

Tinghuan Chen (Member, IEEE) received the B.Eng. and M.Eng. degrees in electronics engineering from Southeast University, Nanjing, China, in 2014 and 2017, respectively, and the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong, in 2021.

He is currently an Assistant Professor with the School of Science and Engineering, The Chinese University of Hong Kong (Shenzhen), Shenzhen, China. His research interests include machine learning for EDA and deep learning accelerators.

Tsung-Yi Ho (Fellow, IEEE) received the Ph.D. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2005.

He is currently a Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Prof. Ho was a recipient of the Best Paper Award at IEEE TCAD in 2015. He currently serves as the VP Conference dor IEEE CEDA and the Executive Committee of ASP-DAC and ICCAD. He is a Distinguished Member of ACM.

Bei Yu (Senior Member, IEEE) received the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Associate Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Dr. Yu received the Eleven Best Paper Awards from top conferences/journals, such as ICCAD 2024, 2021, and 2013, IEEE TSM 2022, DATE 2022, ASPDAC 2021 and 2012, ISPD 2017, and many other awards, including the DAC Under-40 Innovator Award, the IEEE CEDA Ernest S. Kuh Early Career Award, and the Hong Kong RGC Research Fellowship Scheme Award.