Performance-driven Analog Routing via Heterogeneous 3DGNN and Potential Relaxation*

Peng Xu
Chinese University of Hong Kong

Guojin Chen
Chinese University of Hong Kong

Keren Zhu
Chinese University of Hong Kong

Tinghuan Chen
CUHK-Shenzhen

Tsung-Yi Ho
Chinese University of Hong Kong

Bei Yu
Chinese University of Hong Kong

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 paper 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 40nm technology node demonstrate the superiority of the proposed framework compared to the cutting-edge works.

1 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 into three primary categories: template-based [3, 4], simulation-based [5, 6], and heuristic constraint-based approaches [7–10]. 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. Ou et al. expanded upon this constraint by introducing different levels of geometric matching constraints [7]. 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 methods summarize the experience from existing routing solutions, but suffering from scarcity of manual data [11] and lack of explicit performance consideration [12]. GeniusRoute [11] 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 [13], analog routing guidance generation faces challenges related to discreteness, sparsity, and relatively low resolution in 3D 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 [14, 15], we propose the **AnalogFold** framework, which utilizes a potential function to predict routing guidance for individual nets instead of generating a complete routing guidance map.

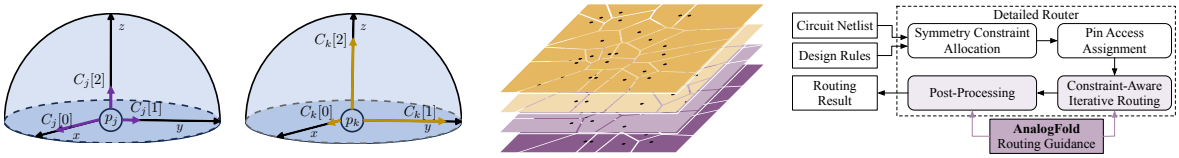
The main contributions of this paper are listed as follows:

- We introduce a performance-driven analog routing approach, which learns from the automatically generated routing patterns using their performance metrics.
- A non-uniform routing guidance is proposed to effectively address sparsity issues by assigning routing guidance to different nets.

*This work is partially supported by The Research Grants Council of Hong Kong SAR (No. CUHK14210723) and The National Natural Science Foundation of China (No. 62304197).

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06.
https://doi.org/10.1145/3649329.3658480



(a) (b) (c)

Figure 1: (a) Two examples of non-uniform routing guidance; (b) The 3D visualization of the non-uniform routing guidance, with each point as a pin access; (c) The overall flow of our performance-driven analog routing method.

- We proposed a customized AnalogFold framework to enable accurate modeling of the performance potential of routing guidance. AnalogFold contains a heterogeneous routing graph, a protein-inspired 3DGNN network, and a pool-aided potential relaxation process.
- 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 3671.00μV, 30.33dB, 169.2MHz, 38.141dB and 2028μV_{rms} improvement in Offset Voltage, CMRR, BandWidth, DC Gain, Noise metrics.

2 Problem Formulation

Analog routing mainly involves placed devices, self-symmetry nets, symmetry net pairs, and design rules. The objective is to route all the nets and optimize post-layout performance. The problem can be formulated as follows:

Problem 1 (Analog Detailed Routing)**.** Given a collection of placed devices $M = \{m_i | 1 \le i \le |M|\}$, a set of self-symmetry nets $N^{SS} = \left\{n_i^{SS} \mid 1 \le i \le \left|N^{SS}\right|\right\}$, a set of nets $N = \{n_i | 1 \le i \le |N|\}$, a set of special nets with specific types $\{N_i^T | 1 \le i \le |N_i^T|\}$, a group of symmetry net pairs $N^{SP} = \left\{n_i^{SP} \mid 1 \le i \le \left|N^{SP}\right|\right\}$, and the design rules, the task is to route all the nets and optimize the post-layout performance.

3 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, AnalogFold divides it into two sub-problems: non-uniform routing guidance generation and guided analog detailed routing.

3.1 Non-uniform Routing Guidance Generation

GeniusRoute uses a uniform 2D 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 2D 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 non-uniform adaptive cost guidance that assigns each pin a different cost guidance, as shown in Figure 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 Figure 1(b). This approach effectively addresses sparsity issues in the route layout by cost guidance to different nets and inherently supports a 3D cost map.

The problem of generating non-uniform routing guidance can be defined as follows:

Problem 2 (Non-Uniform Routing Guidance Generation.)**.** Given the placement of components in the circuit (M), a set of nets (N^*) belonging to the set of nets (N) with specific types (N^T), the objective is to predict a cost guidance set $\mathcal{C}$, where each cost guidance $\mathbf{C}_i$ is of size 1×3. Each element $\mathbf{C}_i[d], d \in \{0, 1, 2\}$ represents the inferred cost guidance of the net N_i is likely the corresponding directions d.

3.2 Guided Analog Routing

Given the circuit netlist, design rules, and performance-driven routing guidance, the guided analog detailed routing problem is defined as:

Problem 3 (Guided Analog Detailed Routing.)**.** Guided analog detailed routing takes placement M, nets N, design rules, and a set of generated non-uniform routing guidance $\mathcal{C} = \{\mathbf{C}_i | 1 \le i \le |N|\}$ as input, and routes all the nets while honoring symmetric constraints and routing guidance with optimal post-layout performance.

Definition 1 (Pin Access Point)**.** In this paper, pin access points refer to the intersections between pin geometry and routing grids. Each pin has at least one access point.

As shown in Figure 1(c), AnalogFold leverages the routing guidance $\mathcal{C}$ generated by machine learning 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 [16], honoring design rules, symmetric constraints, and input non-uniform routing guidance. During each iteration, routing guidance $\mathcal{C}$ 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 [11].

To achieve optimal routing solutions, the key issue is to generate efficient routing guidance generation. We propose AnalogFold for effective 3D routing guidance generation in Section 4.

4 AnalogFold: Analog Performance Modeling and Routing Guidance Prediction

In this section, we present the proposed AnalogFold method flow and detail the algorithms. Figure 2(a) and Figure 2(b) show the overall flow of our routing guidance generation algorithms. Our Analog framework differs from traditional analog methods that rely on

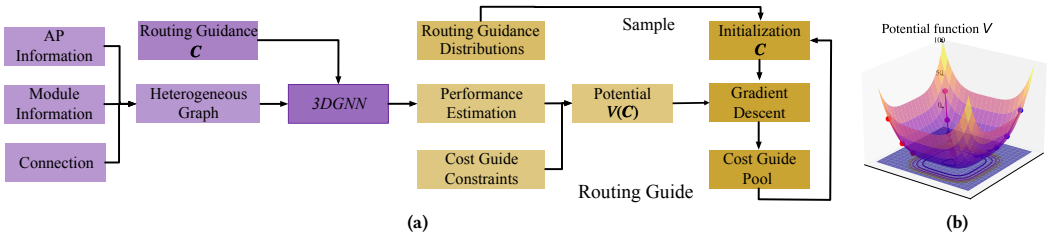


Figure 2: (a) The overview of our AnalogFold, which trains a 3DGNN to model the potential function of routing guidance and uses relaxation to derive the optimized routing guidance; (b) Optimize routing guidance via potential relaxation.

summarizing and learning from human analog routing solutions. Instead, we explicitly construct a performance potential model and use gradient descent to optimize a series of routing guidance for the target design. However, there are several major algorithmic challenges in enabling this methodology:

- **Heterogeneous Information Fusion:** The analog routing map contains both logical and physical connection information. To effectively fuse these two different types of information, we introduce our heterogeneous graph construction for analog routing.
- **Effective Representation Learning in 3D Space:** To tackle the challenge of representing analog routing guidance information in 3D space, we propose the 3DGNN model. It addresses the limitations of directly incorporating coordinate information into features by considering cost-related distances.
- **Potential Relaxation with Pool Assistance:** Gradient descent on non-linear functions suffers from getting trapped in local optima. To optimize a set of solutions through noisy restart, we introduce the potential function modeling and pool-assisted relaxation for routing guidance.

4.1 Heterogeneous Graph for Analog Routing

Routing cost maps for global routing can be formulated into graphs naturally since they are formed by connections of pin access points and modules. 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 Figure 3.

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 Figure 3, these edges indicate the physical connection relationships between pin access points.

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.

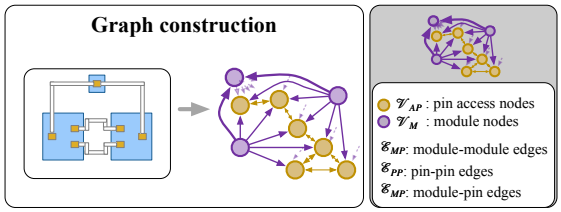


Figure 3: The construction of heterogeneous graph for modeling analog routing.

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.

4.2 Protein-inspired 3DGNN for Analog Routing

Analog routing modeling encounters challenges with discreteness, sparsity, and low resolution compared to placement. The 2D-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 [17] 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.

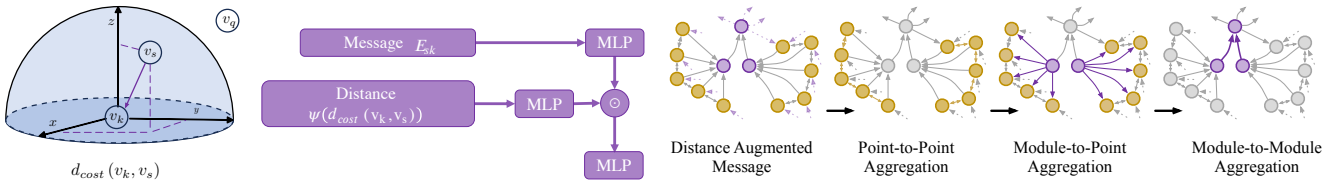


Figure 4: The important components of our proposed 3DGNN model: (a) Routing cost distance; (b) Distance augmented module; (c) Cost-aware message passing procedure on the heterogeneous graph.

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 Figure 4(a). Specifically, we can define the distance honoring routing cost as follows:

$$d_{cost}(v_k, v_s) = \sqrt{(\mathbf{C}_k[0] \cdot h_{ks})^2 + (\mathbf{C}_k[1] \cdot w_{ks})^2 + (\mathbf{C}_k[2] \cdot z_{ks})^2}, \tag{1}$$

where $\mathbf{C}_k$ 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 [17]. With the definition of cost-aware distance, we can define the aggregation function used to encode 3D position information as:

$$\rho^{p\to e}\left(\{\mathbf{r}_h\}_{h=v_k\cup v_k}\right) = \Psi\left(d_{cost}(v_k, v_s)\right) \tag{2}$$

$$= exp(-\gamma||d_{cost}(v_k, v_s) - \mu_k||^2), \tag{3}$$

which converts the position information $\mathbf{r}_h$ to an embedding of distance with radial basis functions. Due to this additional non-linearity, the initial messages are less correlated leading to a faster training procedure.

Cost-aware Message Passing. In 3DGNN, the proposed cost-aware message passing is shown in Figure 4(c), which can be defined as:

$$\begin{aligned} \mathbf{e}_k^l &= \phi^e\left(\mathbf{e}_k, v_{r_k}, v_{s_k}, \mathcal{E}_{s_k}, \rho^{p\to e}\left(\{\mathbf{r}_h\}_{h=r_k\cup s_k}\right)\right), \\ v_i^l &= \phi^v\left(v_i, \rho^{e\to v}\left(\mathcal{E}_i^l\right)\right), \mathbf{u}^l = \phi^u\left(\mathbf{u}, \rho^{v\to u}\left(\mathcal{V}^l\right)\right), \end{aligned} \tag{4}$$

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

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

$$\text{MLP}\left(\text{MLP}\left(v_{r_k}\right) \odot \text{MLP}\left(\Psi\left(d_{cost}(v_k, v_s)\right)\right)\right), \tag{5}$$

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

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

Algorithm 1 Cost-aware Message Passing for 3DGNN

1: **procedure** CAMP
2: $\mathbf{e}_k^l \leftarrow \phi^e\left(\mathbf{e}_k, v_{r_k}, v_{s_k}, \mathcal{E}_{s_k}, \rho^{p\to e}\left(\{\mathbf{r}_h\}_{h=r_k\cup s_k}\right)\right)$. ▷ Update
3: $v_i^l \leftarrow \phi^v\left(v_i, \rho^{e\to v}\left(\mathcal{E}_i^l\right)\right)$. ▷ Aggregate
4: $\mathbf{u}^l \leftarrow \phi^u\left(\mathbf{u}, \rho^{v\to u}\left(\mathcal{V}^l\right)\right)$. ▷ Combine
5: **end procedure**

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:

$$\hat{\mathbf{y}}_{\text{OV}},\ \hat{\mathbf{y}}_{\text{CMRR}},\ \hat{\mathbf{y}}_{\text{UGB}},\ \hat{\mathbf{y}}_{\text{Gain}},\ \hat{\mathbf{y}}_{\text{Noise}} = f_\theta(\mathcal{G}_H, \mathcal{C}), \tag{6}$$

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

4.3 Routing Performance Potential Modeling and Relaxation

Potential Modeling. 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 Figure of Merit (FoM) results of the predicted performance and constraint penalty.

Formally, the potential function associated with the post-layout performance related to routing guidance can be defined as follows:

$$V(\mathcal{C}) = \mathbf{w}_{FoM} \cdot f_\theta(\mathcal{G}_H^{val}, \mathcal{C}) + g(\mathcal{C}), \tag{7}$$

where $\mathbf{w}_{FoM}$ represents the weight preferences for different metrics corresponding to a given FoM, and $g(\mathcal{C})$ describes the constraint function related to the routing guidance. In practice, equal weighting for all terms in FoM led to the best results.

Table 1: 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

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

$$g(\mathrm{C}_i) = -r \sum_{j=1}^{3} (\log \mathrm{C}_i[j] + \log(c_{max} - \mathrm{C}_i[j])), \tag{8}$$

where r is a small positive value that ensures when x approaches the boundary of the feasible region D, $g(\mathrm{C}_i) \to +\infty$, and when x is within D, $g(\mathrm{C}_i) \approx 0$. The feasible region D of Equation (8) 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 $\mathcal{C}_{\text{init}}$ sampled from a given distribution, we can minimize $V(\mathcal{C})$ 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 $p_{relax} \cdot N_{pool}$ 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.

5 Experiments

5.1 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 [18]. The 3dgnn training and cost guide relaxation are implemented with torch-2.0.0 [19]. All designs are in TSMC 40nm technology, extracted for parasitics with Calibre PEX, and simulated with Cadence Spectre.

Benchmarks. Due to significant differences in performance metrics and layout design space of different analog functional building modules, we restrict our research scope to the design of operational transconductance amplifiers (OTAs) as in [11]. OTAs are a classic circuit design in analog and have a wide range of applications. The statistics of our benchmarks are shown in Table 1. We conduct the experiments on two 2-stage miller-compensated operational transconductance amplifiers (OTA1 and OTA2) and two telescopic operational transconductance amplifiers (OTA3 and OTA4). OTA1 and OTA2 have the same circuit topology but different sizing, and the same goes for OTA3 and OTA4. All generated layouts are LVS clean. A/B/C/D represents placements of different net weights.

Compared Methods: In our experiments, we compare our AnalogFold framework with the following strong baselines: 1) MagicalRoute: a default router of the MAGICAL [18] framework proposed in [16]. and 2) GeniusRoute: [11]. MagicalRoute [16] incorporates industrial design rules and analog-specific geometric and electrical constraints to enhance the overall circuit performance. GeniusRoute [11] is the recent work on using machine learning models to guide the analog routing process with generation model VAE, which is the closest one to our work.

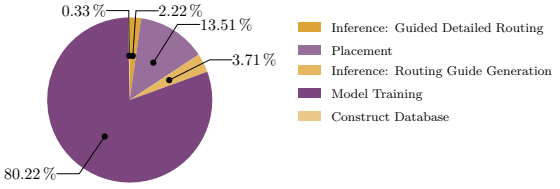
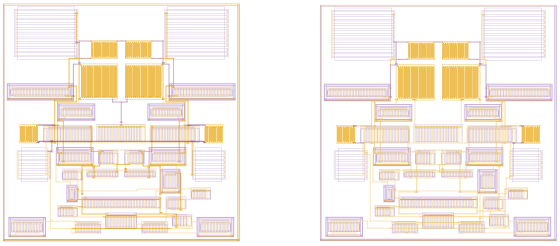


Figure 5: Runtime breakdown for OTA1.



(a) (b)

Figure 6: GeniusRoute and AnalogFold routing solutions.

Evaluation Metrics. The proposed approaches aim to boost the ability to obtain analog circuit layouts with optimal performance. The well-known post-layout metrics (Offset Voltage, CMRR, DC Gain, Bandwidth, 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 AnalogFold.

5.2 Experimental Results

We use the default analog placer in the MAGICAL framework [18] 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 2 demonstrate that our proposed AnalogFold method obtains routing solutions that are superior to baseline methods in terms of post-layout performance. As shown in Table 2, AnalogFold achieves up to 3671.00μV, 30.33dB, 169.2MHz, 38.141dB and 2028μV_{rms} improvement compared to the GeniusRoute [11] concerning Offset Voltage, CMRR, BandWidth, DC Gain, and Noise. On average, the CMRR, Bandwidth, and DC Gain can obtain 16.3%, 11.3%, and 136.8% improvement over the MagicalRoute [16] respectively, while the Offset Voltage and Noise can achieve 45.4% and 21.3% reduction.

Our model exhibits enhanced stability by considering the potential post-layout performance. This is especially evident in the corner case scenario of OTA4-A, where both the optimization-based MagicalRoute [16] and the experience-based GeniusRoute [11] 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 169.2MHz and a gain of 38.14dB.

Table 2: The comparisons between baseline methods and the proposed method.

Circuits		Schematic	[16]	[11]	**Ours**
OTA1-A	Offset Voltage(μV) ↓	-	1466	1464	**1319**
	CMRR(dB) ↑	155.30	89.23	89.01	**112.9**
	BandWidth(MHz) ↑	108.10	50.97	50.97	**51.02**
	DC Gain(dB) ↑	37.19	36.79	36.79	**36.83**
	Noise(μV_{rms}) ↓	213.40	2254	2254	**226**
	Runtime(s) ↓	-	2.49	18.24	5.89
OTA1-B	Offset Voltage(μV) ↓	-	1148	1121	**1031**
	CMRR(dB) ↑	155.30	85.0	85.97	**116.3**
	BandWidth(MHz) ↑	108.10	48.65	48.64	**48.65**
	DC Gain(dB) ↑	37.19	36.58	36.59	**36.62**
	Noise(μV_{rms}) ↓	213.40	2279	2279	**228**
	Runtime(s) ↓	-	2.40	18.30	5.91
OTA1-C	Offset Voltage(μV) ↓	-	405	70.61	**47.47**
	CMRR(dB) ↑	155.30	94.51	93.56	**101.9**
	BandWidth(MHz) ↑	108.10	**49.87**	49.87	49.86
	DC Gain(dB) ↑	37.19	36.89	36.91	**36.92**
	Noise(μV_{rms}) ↓	213.40	2256	**2253**	2254
	Runtime(s) ↓	-	17.65	23.16	6.92
OTA2-A	Offset Voltage(μV) ↓	-	6756	7032	**3361**
	CMRR(dB) ↑	37.72	14.46	13.94	**22.06**
	BandWidth(MHz) ↑	87.54	34.61	34.36	**35.68**
	DC Gain(dB) ↑	34.26	15.04	14.6	**21.71**
	Noise(μV_{rms}) ↓	213.60	426.8	439.6	**303.3**
	Runtime(s) ↓	-	3.51	18.98	8.64
OTA2-B	Offset Voltage(μV) ↓	-	3718	3562	**3135**
	CMRR(dB) ↑	37.72	20.48	20.97	**22.5**
	BandWidth(MHz) ↑	87.54	35.31	35.40	**35.58**
	DC Gain(dB) ↑	34.26	20.45	20.87	**22.15**
	Noise(μV_{rms}) ↓	213.60	319.6	313.8	**299.4**
	Runtime(s)	-	8.34	18.98	7.82
OTA2-C	Offset Voltage(μV) ↓	-	7691	7590	**7558**
	CMRR(dB) ↑	37.72	39.94	39.79	**40.1**
	BandWidth(MHz) ↑	87.54	37.81	37.80	**38.03**
	DC Gain(dB) ↑	34.26	33.51	33.53	**33.63**
	Noise(μV_{rms}) ↓	213.60	264.4	264.7	**265.4**
	Runtime(s) ↓	-	3.29	24.60	16.08
OTA3-A	Offset Voltage(μV) ↓	-	29.68	29.83	**0.249**
	CMRR(dB) ↑	126.60	93.14	93.35	**110.7**
	BandWidth(MHz) ↑	590.00	269.00	267.6	**396.90**
	DC Gain(dB) ↑	47.50	6.066	6.066	**47.66**
	Noise(μV_{rms}) ↓	3061.0 0	2313	2295	**2278**
	Runtime(s) ↓	-	0.47	17.10	7.73
OTA3-B	Offset Voltage(μV) ↓	-	204.4	230.8	**210.6**
	CMRR(dB) ↑	126.60	46.96	46.94	**46.97**
	BandWidth(MHz) ↑	590.00	49.72	51.42	**51.75**
	DC Gain(dB) ↑	47.50	6.863	6.860	**6.949**
	Noise(μV_{rms}) ↓	3061.0	**1570**	1685	1693
	Runtime(s) ↓	-	0.44	15.81	7.48
OTA4-A	Offset Voltage(μV) ↓	-	0.2532	24.14	**0.003**
	CMRR(dB) ↑	110.50	125.4	**139.1**	116.3
	BandWidth(MHz) ↑	802.10	295.0	294.8	**464.0**
	DC Gain(dB) ↑	44.20	6.079	6.079	**44.22**
	Noise(μV_{rms}) ↓	3430.00	2441	2448	**2343**
	Runtime(s) ↓	-	0.66	18.10	7.59
OTA4-B	Offset Voltage(μV) ↓	-	6.783	11.17	**1.137**
	CMRR(dB) ↑	110.50	105.3	93.34	**123.5**
	BandWidth(MHz) ↑	802.10	**304.4**	303.5	304.0
	DC Gain(dB) ↑	44.20	6.078	6.078	**6.078**
	Noise(μV_{rms}) ↓	3430.00	2574	**2551**	2557
	Runtime(s) ↓	-	0.46	18.54	7.52
Average	Offset Voltage(μV) ↓	-	1.000	10.426	**0.546**
	CMRR(dB) ↑	-	1.000	0.998	**1.163**
	BandWidth(MHz) ↑	-	1.000	1.002	**1.113**
	DC Gain(dB) ↑	-	1.000	0.999	**2.368**
	Noise(μV_{rms}) ↓	-	1.000	1.007	**0.787**
	Runtime(s) ↓	-	**1.000**	17.147	7.480

Figure 5 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 [16], it is nearly 2.29× faster than GeniusRoute [20] due to the simplified 3D graph structure.

6 Conclusion

In this paper, we present a new paradigm in analog IC routing which learns routing guidance from solutions by an automatic routing engine. Our approach, AnalogFold, combines non-uniform 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. Experimental results show that our framework significantly improved multiple post-layout simulation metrics.

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.

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

[3] J. Crossley, A. Puggelli, H.-P. Le, B. Yang, R. Nancollas, K. Jung, L. Kong, N. Narevsky, Y. Lu, N. Sutardja *et al.*, "BAG: A designer-oriented integrated framework for the development of ams circuit generators," in *Proc. ICCAD*, 2013.

[4] E. Chang, J. Han, W. Bae, Z. Wang, N. Narevsky, B. Nikolic, and E. Alon, "BAG2: A process-portable framework for generator-based ams circuit design," in *Proc. CICC*, 2018.

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

[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.

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

[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.

[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.

[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.

[11] K. Zhu, M. Liu, Y. Lin, B. Xu, S. Li, X. Tang, N. Sun, and D. Z. Pan, "GeniusRoute: A new analog routing paradigm using generative neural network guidance," in *Proc. ICCAD*, 2019.

[12] H. Chen, K.-C. Hsu, W. J. Turner, P.-H. Wei, K. Zhu, D. Z. Pan, and H. Ren, "Reinforcement learning guided detailed routing for custom circuits," in *Proc. ISPD*, 2023.

[13] M. Liu, K. Zhu, J. Gu, L. Shen, X. Tang, N. Sun, and D. Z. Pan, "Towards decrypting the art of analog layout: Placement quality prediction via transfer learning," in *Proc. DATE*, 2020.

[14] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. Nelson, A. Bridgland *et al.*, "Improved protein structure prediction using potentials from deep learning," *Nature*, vol. 577, no. 7792, pp. 706–710, 2020.

[15] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[16] 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.

[17] K. Schütt, P.-J. Kindermans, H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller, "Schnet: A continuous-filter convolutional neural network for modeling quantum interactions," in *Proc. NIPS*, 2017.

[18] H. Chen, M. Liu, X. Tang, K. Zhu, A. Mukherjee, N. Sun, and D. Z. Pan, "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.

[19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Proc. NIPS*, vol. 32, 2019.

[20] 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.