VinePlace: Compact Analog and Mixed-Signal Placement via Core-Centered Vine Representation

Weijian Fan[1,2], Zhiyuan Zhao[3], Jingpeng Wang[4], Haoyi Zhang[2], Jincheng Lou[2],
Runsheng Wang[2,5,6], Xiyuan Tang[4], Yibo Lin[2,5,6*]

{[1]School of Software and Microelectronics, [2]School of Integrated Circuits, [3]School of EECS, [4]Institute for Artificial Intelligence,}, Peking University, [5]Beijing Advanced Innovation Center for Integrated Circuits,
[6]Institute of Electronic Design Automation, Peking University, Wuxi, China

Abstract

Analog and mixed-signal (AMS) placement is a critical stage in custom integrated circuit design, and its automation has become indispensable as circuit scales continue to grow. However, existing analog placement methods often lack an explicit layout organization centered on performance-sensitive modules and tend to leave excessive whitespace, making it difficult to achieve compact and high-performance layouts. To address this issue, we propose *VinePlace*, a placement framework for AMS circuits built on a novel core-centered vine representation and its hierarchical extensions for diverse AMS structures. Based on this representation, VinePlace further incorporates geometry-level refinement and hierarchical packing to generate compact legal placements while preserving competitive post-layout performance. Experimental results on five benchmarks, ranging from a 17-device two-stage OTA to a 1,623-device SAR ADC, show that VinePlace achieves 1.03×–1.52× area reduction and 1.07×–1.71× HPWL reduction over the best-performing baselines, generating highly compact layouts with excellent post-layout performance. On the large-scale SAR ADC benchmark, VinePlace further demonstrates good scalability and generates high-quality layouts in an automated flow.

1. Introduction

Analog and mixed-signal (AMS) layout synthesis remains a critical and time-consuming stage of custom integrated circuit design [1]. Unlike highly automated digital flows, analog placement must satisfy symmetry, matching, proximity, and boundary constraints [2–8], while implementation parasitics directly affect post-layout performance, area, and routability [9–12]. The growing scale and hierarchical complexity of AMS systems make manual placement increasingly difficult, making automatic constraint-aware placement an important problem in Electronic Design Automation (EDA).

Prior works on analog placement can be broadly categorized into three directions: machine-learning-based methods, analytical optimization methods, and representation-based methods.

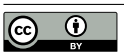
Machine-learning-based methods have recently attracted increasing attention for analog placement, including performance prediction, placement guidance, and direct learning-based generation. Early studies employed neural networks to estimate layout quality or circuit performance from placement solutions [13–18]. More recent efforts further integrate learning into the placement loop, such as ML-guided well-aware placement and multi-task-learning-based placement generation [19–21]. These methods improve automation capability and may capture complex correlations between layout and circuit performance. However, they still rely heavily on training data and often struggle to generalize across unseen circuit topologies.

Analytical optimization methods provide another important direction by optimizing continuous placement variables through differentiable formulations or gradient-based search. Prior works have incorporated layout-dependent effects [9], device-layer awareness [10], system signal flow and charge-flow guidance [22, 23], hierarchical optimization [11, 24], routability and advanced-node constraints [25–28], and performance-aware optimization considering current flows and pole-related effects [29]. More recent efforts further explore joint optimization for hierarchical AMS placement [12]. These methods are attractive for their optimization efficiency and scalability on larger layouts. However, it remains difficult for them to directly encode diverse analog layout constraints, and they may also be sensitive to initialization and nonconvex optimization landscapes, which can hinder convergence to compact and high-quality placements.

Representation-based methods remain a practical and effective solution for constrained analog placement because they can naturally encode structural constraints and are well suited for generating compact layouts. Classical representations such as Sequence Pair, O-tree, B*-tree, and TCG-S have all been extended to analog placement problems [30–34]. Based on Sequence Pair, Tam et al. incorporated symmetry, boundary, and preplaced constraints into the placement process [2]. Lin et al. proposed a B*-tree-based symmetry-island representation to preserve forced symmetry [3], and later extended this line of work with a hierarchical B-tree framework that simultaneously handles matching, symmetry, and proximity constraints [35]. Tsao et al. proposed CB-tree to better capture module adjacency and support multiple analog constraints [36], while Wu et al. introduced QB-tree to explore feasible placements under diverse constraints using a quadtree structure [37]. These methods are attractive because they preserve structural regularity and naturally work with simulated annealing.

Despite these advantages, existing representation-based methods still have at least three notable limitations: First, they fail to anchor the placement on performance-sensitive modules, potentially degrading post-layout performance. Second, they lack a unified

*Corresponding author: yibolin@pku.edu.cn



ICCAD '26, San Jose, CA, USA

ACM ISBN 979-8-4007-2873-0/2026/11
https://doi.org/10.1145/3831252.3833944

hierarchical representation for modeling large nested AMS circuits. Third, once the global topology converges, they lack an effective mechanism for local geometry refinement, severely restricting further whitespace reduction and area optimization.

To address these limitations, we propose VinePlace, a placement framework for AMS circuits based on a novel vine representation. The main contributions of this paper are summarized as follows:

(1) We propose a novel **core-centered vine representation** and a **hierarchical vine family** covering constrained, unconstrained, multi-group, and nested AMS circuits.
(2) We propose a **geometry-level refinement** mechanism and incorporate it into representation-aware perturbation to enable joint exploration of global topology optimization and local geometry refinement.
(3) We propose a **hierarchical packing method with contour maintenance** that, together with the vine representation and representation-aware perturbations, enables efficient compact layout generation for hierarchical AMS circuits.
(4) Experiments from a **17-device OTA to a 1,623-device SAR ADC** show 1.03×–1.52× area and 1.07×–1.71× wirelength reductions over the best baselines with competitive post-layout quality.

2. Preliminaries

2.1. Problem Definition

For analog circuit placement, the problem can be formulated as minimizing the placement cost subject to layout constraints:

$$\begin{aligned} \min \quad & \textit{Area} + \textit{Wirelength} \\ \text{s.t.} \quad & \textit{Placement Constraints} \end{aligned} \tag{1}$$

where *Area* denotes the total layout area, and *Wirelength* denotes the total half-perimeter wirelength (HPWL). *Placement Constraints* include:

- **Symmetry Constraint**: Some devices must be placed symmetrically or self-symmetrically.
- **Proximity Constraint**: To reduce parasitic effects, the distance between certain components must be below a specified threshold.
- **Matching Constraint**: Matched devices, such as current mirrors and differential pairs, should be placed close to each other to improve electrical matching.
- **Non-overlap Constraint**: No devices should overlap with each other.
- **Boundary Constraint**: All devices must be placed within the prescribed layout boundary.

2.2. Hierarchical B*tree

In high-performance analog placement, modules are often clustered into functional units—such as symmetry groups or proximity groups—to minimize process variations and parasitic effects. The hierarchical B*-tree (HB*-tree) [35] is an advanced extension of the classical B*-tree, specifically designed to handle these complex constraints. It introduces the concept of hierarchical nodes, where a single node in the top-level tree can encapsulate an entire subtree representing a specific constraint group.

To handle symmetry constraints, the HB*-tree utilizes an automatically symmetric-feasible B*-tree (ASF-B*-tree) representation to model symmetry groups. As shown in Figure 1, an ASF-B*-tree consists of multiple symmetric module pairs or self-symmetric modules. To ensure that devices meet the symmetry requirement, the ASF-B*-tree explicitly records only half of the nodes, with the other half being mirrored across the symmetry axis. For self-symmetric modules, the method restricts them to the rightmost branch to prevent the two halves from being separated.

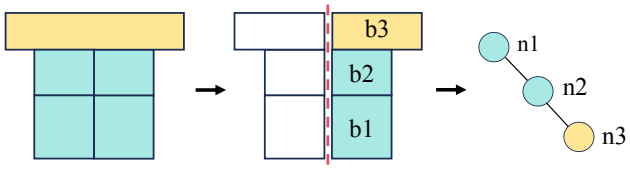


Figure 1: The representation of ASF-B∗-tree.

3. Algorithm

3.1. Overview

VinePlace consists of four components: a vine representation, representation-aware perturbation, hierarchical packing with contour maintenance, and simulated-annealing-based optimization, as shown in Fig. 2. Given an input netlist, layout constraints, and Pcell descriptions, VinePlace initializes a vine topology and assigns parent–child relationships among modules. During optimization, representation-aware perturbations modify the topology and module geometries. Each candidate is converted into a physical placement through hierarchical packing, with contour maintenance tracking the exposed boundaries. The placement is then evaluated and accepted or rejected according to the simulated-annealing criterion. After convergence, the optimized module geometries and coordinates are exported for downstream layout generation.

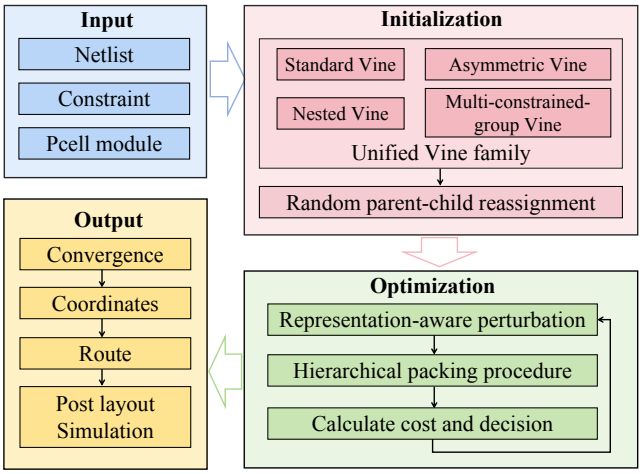


Figure 2: The overview of VinePlace.

3.2. Vine Representation

To support structured AMS placement, we develop a core-centered vine representation inspired by analog layout practice, where performance-sensitive constrained cores are placed first and peripheral modules are attached along their exposed contours. The constrained

core therefore anchors the placement topology, while contour interfaces connect the surrounding modules. Based on this principle, we construct a unified vine family for constrained, unconstrained, multi-group, and nested AMS circuits.

3.2.1. Standard Vine Representation

The standard vine is the basic instantiation of the proposed framework for constrained analog placement. A standard vine consists of three tiers of nodes:

Tier-1 nodes: Constrained-core nodes. Tier-1 nodes represent the structural core of the placement, including differential pairs, current mirrors, and other sensitive symmetry modules. The Tier-1 constrained core is internally represented by an ASF-B*-tree, as shown in Figure 3(a) and Figure 3(b). In analog circuit design, these constrained core structures are typically the most critical to circuit performance, and layout engineers naturally assign them the highest priority during manual placement. In our representation, we explicitly treat them as the anchors of the placement. This approach preserves their structural regularity and prevents sensitive modules from being placed at layout boundaries in an uncontrolled manner, which helps maintain excellent post-layout performance.

Tier-2 nodes: Contour-interface nodes. After the constrained-core module is packed, its contour is extracted and partitioned into upper, lower, left, and right contour branches, as illustrated in Figure 3(a). These contour branches are then represented as Tier-2 nodes in Figure 3(b). Tier-2 nodes are virtual nodes and do not correspond to physical devices; instead, they serve as interfaces that expose legal connection locations surrounding the constrained core. Each contour-interface node can have at most one child node to maintain a clear connection structure.

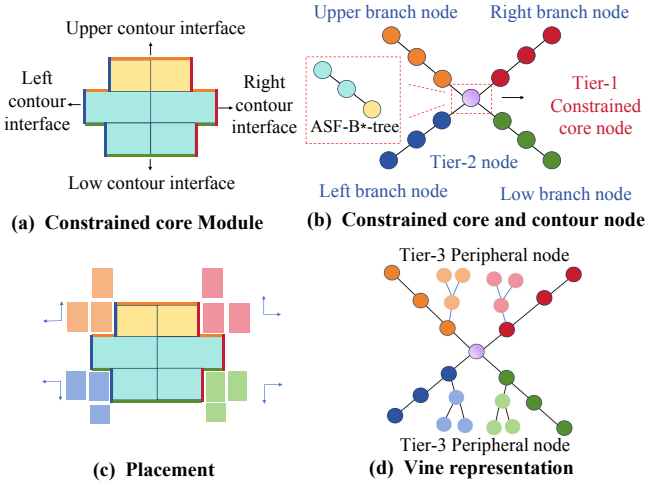
Tier-3 nodes: Peripheral-module nodes. Tier-3 nodes represent the remaining modules attached to the contour-interface nodes. In conventional B*-tree-based structures, child modules are typically attached through rightward or upward expansion. In contrast, Tier-3 nodes can be organized around the constrained core in various directions through contour interfaces. These modules do not belong to the constrained core and are therefore optimized with greater freedom to reduce whitespace and improve placement compactness, as shown in Fig. 3(c) and Fig. 3(d).

Packing follows a strictly hierarchical realization process. The Tier-1 constrained core is packed first, followed by the extraction of contour interfaces to form Tier-2. Finally, the Tier-3 modules are attached and packed via the Tier-2 interfaces. Ultimately, this multi-tier representation provides a significant dual advantage: it strictly preserves the layout regularity of critical analog cores, while giving peripheral modules the freedom to expand in multiple directions, resulting in highly compact and high-performance placements.

3.2.2. A Unified Family of Vine Representations

To support a broader range of analog and hierarchical AMS layouts, we further extend it into a unified vine family based on the same design principle: organizing placement around structural cores, exposing legal contour-based interfaces, and attaching surrounding modules through a hierarchical topology.

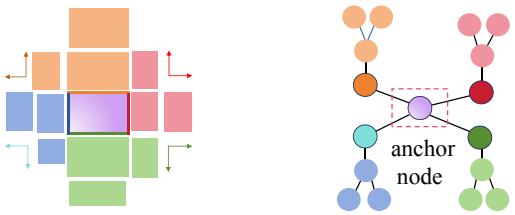
(1) Asymmetric Vine. For circuit blocks without any constrained groups, we employ the Asymmetric Vine. As shown in Fig. 4, the constrained core in Tier-1 is replaced by an unconstrained



(a) Constrained core Module (b) Constrained core and contour node (c) Placement (d) Vine representation

Figure 3: The representation of standard vine.

anchor node, while Tier-2 and Tier-3 retain the same contour-interface and peripheral-connection roles as in the standard vine. To avoid ineffective center selection, the anchor node is selected from the unconstrained modules based on a weighted combination of module area and connectivity. This ensures that the vine is organized around a centrally important module, which helps reduce whitespace and improve layout compactness.

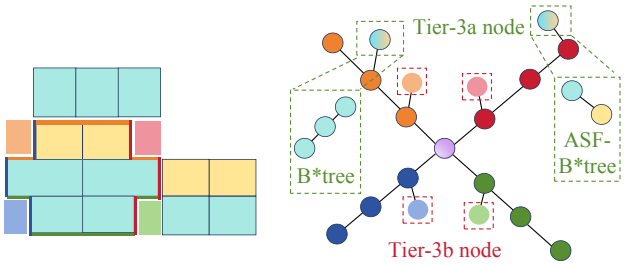


(a) Placement (b) Vine representation

Figure 4: The representation of Asymmetric vine.

(2) Multi-Constrained-Group Vine. The representation is shown in Figure 5. In this representation, one constrained group is chosen as the primary constrained core in Tier-1 nodes, while the remaining constrained groups are incorporated into the outer vine as Tier-3a auxiliary constrained nodes. Tier-3a nodes include not only additional symmetry islands, which are still represented by ASF-B*-trees, but also matching groups and proximity-sensitive module groups, which can be represented by B*-trees. Ordinary peripheral modules are placed in Tier-3b nodes. In this way, the representation preserves the main performance-sensitive core while still allowing auxiliary constrained groups and ordinary modules to be organized within a unified vine hierarchy. The packing process proceeds hierarchically from the Tier-1 constrained core to the Tier-2 contour interfaces, followed by Tier-3a auxiliary constrained nodes and finally Tier-3b peripheral modules.

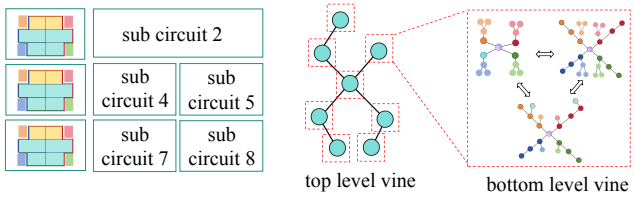
(3) Nested Vine. Large-scale AMS circuits, such as ADCs and DACs, are inherently hierarchical and composed of multiple functional sub-circuits. To capture this structure, we introduce the



(a) Placement (b) Vine representation

Figure 5: The representation of multi-constrained-group vine.

Nested Vine, as shown in Fig. 6. Each sub-circuit is first represented by a bottom-level vine from the vine family according to its constraint type, while a top-level vine is constructed to organize the resolved sub-circuits as higher-level modules. In this way, the proposed representation is extended recursively from local constrained structures to system-level hierarchy. During simulated annealing, the topological search explores the state space of both bottom-level and top-level vines jointly. In contrast, the packing process is performed in a strictly bottom-up manner: each sub-vine is first packed locally, and the resulting sub-circuits are then treated as modules for top-level packing. This separation between hierarchical search and bottom-up packing enables the vine framework to handle hierarchical AMS layouts beyond single-block analog placement.



(a) Placement (b) Vine representation

Figure 6: The representation of nested vine.

3.3. Representation-Aware Perturbation Operators

Once initialized, VinePlace explores the placement space using *representation-aware perturbations*. Topology-level perturbations optimize the global vine hierarchy while preserving validity, whereas geometry-level refinement reduces whitespace through transistor finger reconfiguration. At each simulated-annealing iteration, one operator is applied and the resulting placement is evaluated.

3.3.1. Topology-Level Perturbations

(1) Constrained-Preserving Perturbations. Constrained-preserving perturbations are applied to constrained groups, including Tier-1 constrained-core nodes and Tier-3a auxiliary constrained nodes. Their purpose is to explore the topological organizations while preserving the internal regularity of symmetry-, matching-, or proximity-constrained groups. Three basic perturbations are supported: **(1) move**, which inserts a node into another valid position within its tree; **(2) swap**, which exchanges the positions of two nodes; and **(3) rotate**, which changes the aspect ratio of a node by swapping its width and height when allowed by device geometry.

For symmetry-constrained structures, the above three perturbations are applied to the two symmetric modules simultaneously to maintain their symmetry. For matching groups, while the move and swap operations are applied to individual modules, the rotate operation is strictly applied to the entire group to prevent orientation-induced electrical mismatch. Finally, for proximity groups, all three perturbations are applied to individual modules.

(2) Peripheral Reorganization Perturbations. Peripheral reorganization perturbations are applied to Tier-3b peripheral modules. Their purpose is to reorganize the surrounding modules more flexibly and improve placement compactness.

Three basic perturbations are used: **(1) move**, which relocates a module to another valid contour-branch position; **(2) swap**, which exchanges the positions of two modules within the same branch or across different branches; and **(3) rotate**, which rotates the module when allowed by device geometry.

By performing these perturbations along the four contour branches, peripheral modules can be flexibly reorganized in all directions around the constrained core, thereby significantly expanding the search space.

3.3.2. Geometry-Level Refinement

Beyond topology-level perturbations, VinePlace further introduces a *geometry-level refinement mechanism* based on transistor finger reconfiguration to reduce whitespace area. Unlike perturbations that alter the vine topology, this refinement preserves the total transistor width and directly fine-tunes the module's aspect ratio via finger-number reconfiguration, further compacting the layout area.

This refinement is activated only when topology-level perturbations fail to improve placement compactness for a number of consecutive iterations. At that stage, the current topology is regarded as initially stable, and the remaining inefficiency mainly comes from local geometric mismatch between transistor shapes and contour boundaries. For a selected transistor, the finger number is constrained to remain within a feasible range allowed by the PDK. For devices in symmetry or matching constrained groups, the same finger update is applied to all transistors in the group to preserve placement constraints. VinePlace then explores two neighboring finger configurations, namely $n_f + 1$ and $n_f - 1$, whenever feasible. After each trial, the width and height of the selected module are updated, the placement is repacked, and the cost is recomputed. The neighboring configuration with lower cost is chosen as the candidate perturbation and further evaluated under the simulated annealing acceptance rule.

In this way, finger reconfiguration complements topology perturbation by improving local contour fitting, reducing whitespace, and enhancing placement compactness.

3.4. Hierarchical Packing with Contour Maintenance

3.4.1. Hierarchical Packing

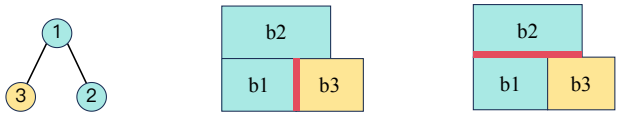
Given a perturbed vine topology, hierarchical packing determines the physical module positions in a center-out order. Tier-1 cores are packed first, followed by Tier-2 contour extraction, internal packing of Tier-3a groups, and outer-level packing of all Tier-3 modules. Contours are updated throughout this process to support subsequent module attachment.

The hierarchical packing process is carried out as follows:

(1) Tier-1 packing. Tier-1 forms the structural core of the placement. In the standard vine, Tier-1 nodes constitute a symmetry island represented by an ASF-B*-tree. Packing starts from one half of the island, and the other half is determined by mirroring with respect to the symmetry axis. As illustrated in Fig. 7, a left child inherits the parent y-coordinate and is placed against the current right contour, whereas a right child inherits the parent x-coordinate and is placed against the current top contour. Specifically,

$$y_{ch} = y_{pa}, x_{ch} = \max(\text{Contour}_{right}(y_{ch},\ y_{ch} + H_{ch})) \quad (2)$$

$$x_{ch} = x_{pa}, y_{ch} = \max(\text{Contour}_{top}(x_{ch},\ x_{ch} + W_{ch})) \quad (3)$$



(a) topology (b) Left children (c) Right children

Figure 7: Tier-1 packing under the vine representation.

For the Asymmetric Vine, Tier-1 is replaced by an unconstrained anchor node, which is first packed as the structural center of placement.

(2) Tier-2 extraction. After Tier-1 is packed, Tier-2 contour-interfa-ce nodes are extracted from the exposed contour of Tier-1 nodes. Since Tier-2 nodes are virtual nodes, they have no physical width or height; only their interface coordinates are stored. For top and bottom contour nodes, the stored form is (x_{start}, x_{end}, y) with $x_{start} < x_{end}$. For left and right contour nodes, the stored form is (x, y_{start}, y_{end}) with $y_{start} < y_{end}$. It should be noted that each Tier-2 node can have at most one child node.

(3) Tier-3a internal packing. For the Multi-constrained-group vine, each Tier-3a auxiliary constrained group is first packed according to its internal representation, and then its overall width and height are extracted for outer-level packing. Specifically, auxiliary symmetry islands are resolved by ASF-B*-trees, while matching/proximity-sensitive groups are resolved by compact B*-trees. After this step, each Tier-3a group is treated as a higher-level module in the outer vine structure.

(4) Outer-level Tier-3 packing. After the Tier-3a groups are internally packed, all Tier-3 nodes, including Tier-3a and Tier-3b, are packed in the outer vine structure. Two cases are considered.

(a) Tier-3 nodes directly attached to Tier-2. For Tier-3 nodes directly attached to Tier-2, the initial coordinates are

$$\text{Top branch: } (x_{ch}, y_{ch}) = (x_{start},\ y_{pa}), \quad (4)$$

$$\text{Bottom branch: } (x_{ch}, y_{ch}) = (x_{end} - W_{ch},\ y_{pa} - H_{ch}), \quad (5)$$

$$\text{Left branch: } (x_{ch}, y_{ch}) = (x_{pa} - W_{ch},\ y_{start}), \quad (6)$$

$$\text{Right branch: } (x_{ch}, y_{ch}) = (x_{pa},\ y_{end} - H_{ch}). \quad (7)$$

(b) Tier-3 nodes attached to existing Tier-3 nodes. For Tier-3 nodes attached to existing Tier-3 nodes, branch-aware local growth rules are summarized in Table 1.

3.4.2. Contour Maintenance

As illustrated in Fig. 7, module coordinates are determined from current contours, making continuous boundary updates essential throughout hierarchical packing. To achieve this, VinePlace uses the unified contour-maintenance procedure across Tier-1, Tier-2, and Tier-3 packing. The algorithm is shown in Algorithm 1.

Table 1: Growth rules for outer-level Tier-3 packing.

Branch	Growth Direction	Queried Contours
Top	left + up	left / top
Bottom	right + down	right / bottom
Right	right + up	right / top
Left	left + down	left / bottom

Taking the top contour as an example, the procedure first extracts the top boundary segments of all packed modules and converts them into entry and exit events sorted by x-coordinate. A sweep-line process is then performed while maintaining an active multiset of segment heights: an entry event inserts the corresponding height, whereas an exit event removes it. For each interval between two consecutive events, the exposed top contour is determined by the maximum active height, and adjacent segments with the same height are merged to obtain the final contour.

The same procedure is applied to the bottom, left, and right contours. Its time complexity is $O(N \log N)$, where N is the number of packed modules.

Algorithm 1 Contour Maintenance

Require: Packed modules $\mathcal{M} = \{m_1, m_2, \ldots, m_N\}$, direction d
Ensure: Contour set C_d

```
Extract boundary segments S_d = {s_1, s_2, ..., s_K} from M
Convert each s_i ∈ S_d into an entry event e_i^in and an exit event e_i^out
Sort the event set E_d = {e_1, e_2, ..., e_2K} along direction d
Initialize A_d ← ∅ and C_d ← ∅
for n = 1 to 2K − 1 do
    (x_n, y_n, t_n) ← e_n
    Update A_d according to the event type t_n
    if x_n ≠ x_{n+1} and A_d ≠ ∅ then
        h_n ← max(A_d)
        c_n ← (x_n, x_{n+1}, h_n)
        C_d ← C_d ∪ {c_n}
    end if
end for
Merge adjacent segments in C_d with the same height
return C_d
```

3.5. Cost and Optimization Process

3.5.1. Cost Function

In VinePlace, most placement constraints are enforced by the vine representation and hierarchical packing, rather than being added as penalty terms in the objective function. Specifically, symmetry is preserved in Tier-1 and Tier-3a packing, while matching and proximity are maintained in Tier-3a through the structural organization. In addition, performance-sensitive modules are placed around the constrained central region, which helps reduce the impact of peripheral parasitics and long interconnects on post-layout performance.

With these hard constraints handled during representation construction and packing, the cost function is defined as a weighted

sum of normalized layout area and normalized half-perimeter wirelength (HPWL):

$$\text{Cost} = \alpha \cdot \widehat{\text{Area}} + \beta \cdot \widehat{\text{HPWL}},$$

where $\widehat{\text{Area}} = \text{Area}/\text{Area}_0$ and $\widehat{\text{HPWL}} = \text{HPWL}/\text{HPWL}_0$. Here, Area and HPWL denote the layout area and HPWL of the current solution, respectively, while Area_0 and HPWL_0 are the corresponding values of the initial placement. In all experiments, we set $\alpha = \beta = 1$ and use the same values across all benchmarks. This fixed equal-weight setting provides a simple and uniform objective without benchmark-specific weight tuning.

The optimization is performed using simulated annealing. A perturbation with $\Delta\text{Cost} \leq 0$ is always accepted; otherwise, it is accepted with probability $P = \exp(-\Delta\text{Cost}/T)$. Across all benchmarks, the simulated annealing process starts from $T_0 = 1$ and terminates at $T_{\min} = 0.01$.

3.5.2. Optimization Process

The overall optimization flow of VinePlace is summarized in Algorithm 2. Starting from an initialized vine topology, VinePlace iteratively applies representation-aware perturbation operators, performs hierarchical packing, evaluates the placement cost, and accepts or rejects candidate solutions under the simulated annealing criterion until the temperature reaches $T_{\min}$.

After optimization, the final module geometries and placement coordinates are exported for downstream layout generation. Resized devices are regenerated using a SKILL-based Pcell generator, and the optimized placement is then used for placement and routing to produce the final GDSII layout for post-layout evaluation.

Algorithm 2 VinePlace Optimization Flow

Require: Netlist $\mathcal{L}$, constraints C, Pcell library $\mathcal{P}$
Ensure: Optimized placement $\mathcal{X}^*$

Initialize vine topology $\mathcal{V}$
$\mathcal{X} \leftarrow \textsc{HierarchicalPacking}(\mathcal{V})$
$\mathcal{V}^* \leftarrow \mathcal{V}, \mathcal{X}^* \leftarrow \mathcal{X}, T \leftarrow T_0$
while $T > T_{\min}$ **do**
  if GeometryRefinementEnabled **then**
    $O \leftarrow \textsc{SelectOperator}(O_{\text{topo}} \cup O_{\text{geo}})$
  else
    $O \leftarrow \textsc{SelectOperator}(O_{\text{topo}})$
  end if
  $\mathcal{V}' \leftarrow \textsc{ApplyOperator}(\mathcal{V}, O)$
  $\mathcal{X}' \leftarrow \textsc{HierarchicalPacking}(\mathcal{V}')$
  $\Delta\text{Cost} \leftarrow \text{Cost}(\mathcal{X}') - \text{Cost}(\mathcal{X})$
  if $\Delta\text{Cost} \leq 0$ **or** $\exp(-\Delta\text{Cost}/T) > \text{rand}(0,1)$ **then**
    $\mathcal{V} \leftarrow \mathcal{V}', \mathcal{X} \leftarrow \mathcal{X}'$
    if $\text{Cost}(\mathcal{X}) < \text{Cost}(\mathcal{X}^*)$ **then**
      $\mathcal{V}^* \leftarrow \mathcal{V}, \mathcal{X}^* \leftarrow \mathcal{X}$
    end if
  end if
  $T \leftarrow \textsc{UpdateTemperature}(T)$
end while
Export optimized module geometries and placement coordinates
return $\mathcal{X}^*$

4. Experimental Results

The VinePlace framework is implemented in C++ and integrated into the PANDA analog design framework [38][1], where it serves as the placement engine in the end-to-end design flow. All experiments are conducted on a workstation equipped with a 24-core Intel Core Ultra 9 275HX CPU @ 2.70 GHz. The constraints for all benchmarks are pre-defined by domain experts. During the routing stage, an A*-based analog router SAGERoute [39] is employed. To obtain post-layout results, we utilize Cadence Spectre, and Calibre. Specifically, Calibre PEX is configured to extract parasitic resistance, parasitic capacitance, and coupling capacitance (R+C+CC mode). To ensure a fair comparison, all compared methods use the same routing, parasitic extraction, and post-layout evaluation flow throughout the experiments.

4.1. Benchmark

All circuits in our work are implemented using a 65nm CMOS process technology. We evaluate our framework across five analog circuits: a 17-device Miller two-stage operational transconduct amplifier (OTA), a 24-device comparator, a 53-device Voltage-Controlled Oscillator (VCO), a 49-device three stage OTA, and a large-scale successive approximation register analog-to-digital converter (SAR ADC) containing 1,623 devices. The schematics of the five circuits are shown in Figure 8. The corresponding device and net counts are summarized in Table 2.

Table 2: Scale of the five benchmark circuits

Circuit	Device	Net
Two stage OTA	17	13
Comparator	24	17
VCO	53	23
Three stage OTA	49	33
SAR ADC	1623	709

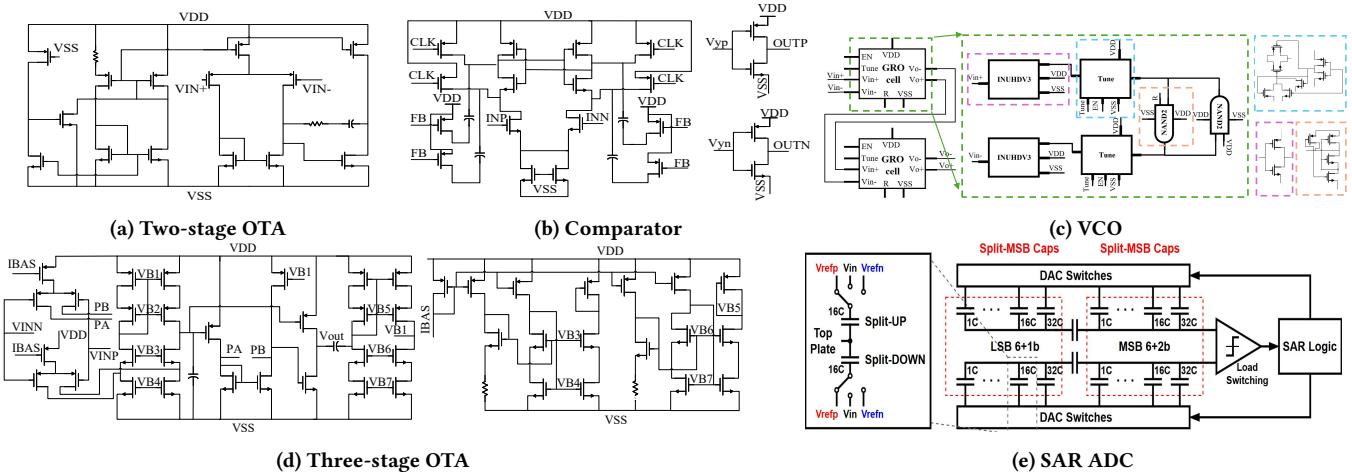
4.2. Experiment Results

We compare VinePlace against three representative baselines: JointPlace [12], MAGICAL [40], and HBTree [35]. JointPlace is a recent AMS placer for hierarchical AMS circuits. MAGICAL is an open-source full-flow analog layout framework that integrates placement and routing. HBTree is a topology-based analog placement algorithm. For all benchmarks, pre-layout schematic results are treated as the golden reference. We compare post-layout performance, layout area, and HPWL across all methods. The improvement ratios of area and HPWL are computed with respect to the best-performing baseline for each benchmark.

4.2.1. Two-Stage OTA

On the two-stage OTA benchmark, VinePlace achieves the best overall trade-off between post-layout performance and physical compactness among the compared automated methods. It delivers the highest gain (43.85 dB), phase margin (72.10°), and GBW (138.81 MHz), while also obtaining the smallest layout area of 3883 μm^2 and the shortest HPWL of 87 μm. In terms of physical metrics, VinePlace achieves 1.52× and 1.60× improvements in area and HPWL, respectively, over the best competing automated baseline.

[1]PANDA is available at https://github.com/PKU-IDEA/PANDA.



(a) Two-stage OTA (b) Comparator (c) VCO

(d) Three-stage OTA (e) SAR ADC

Figure 8: The schematics of five benchmarks.

Table 3: Experimental comparison of VinePlace and other methods on five benchmark circuits.

Method	Two-Stage OTA						Comparator					VCO				
	Gain (dB)	PM (°)	GBW (MHz)	SR (V/μs)	Area (μm^2)	HPWL (μm)	V_{off} (μV)	T_{dly} (ns)	V_{ns} (μV)	Area (μm^2)	HPWL (μm)	Freq. (GHz)	PN (dBc)	K_{vco} (GHz/V)	Area (μm^2)	HPWL (μm)
Schematic	47.13	73.71	176.87	450.56	–	–	20.52	0.927	104.5	–	–	4.95	-80.7	2.51	–	–
Magical[40]	37.20	71.28	132.62	195.73	6211	244	289.2	0.956	148.2	1653	218	2.47	-87.5	1.40	1632	351
JointPlace[12]	42.72	70.01	124.65	**222.27**	8302	234	264.5	1.005	117.5	1081	201	2.66	**-88.2**	1.47	1317	301
HBTree[35]	35.42	70.34	100.65	189.30	5906	139	199.2	0.979	150.1	897	132	2.52	-85.0	1.49	866	327
VinePlace	**43.85**	**72.10**	**138.81**	197.43	**3883**	**87**	**180.5**	**0.939**	**115.0**	**834**	**77**	**2.85**	-85.7	**1.59**	**718**	**282**
Ratio	–	–	–	–	**1.52×**	**1.60×**	–	–	–	**1.08×**	**1.71×**	–	–	–	**1.21×**	**1.07×**

Three stage OTA							SAR ADC						
Method	Gain (dB)	PM (°)	UGB (MHz)	Noise (μV)	Area (μm^2)	HPWL (μm)	Method	SNDR (dB)	ENOB (bit)	P_{core} (μW)	FOM (fJ)	Area (μm^2)	HPWL (μm)
Schem.	79.07	66.10	10.51	132.6	–	–	**Schem.**	69.60	11.27	297.1	4.823	–	–
Magical[40]	77.52	47.32	5.872	132.7	8764	1062	**Human**	68.63	11.11	348.3	6.318	20795	5482
JointPlace[12]	77.56	46.37	6.317	125.9	6730	902	**JointPlace[12]**	**67.60**	**10.94**	369.3	7.544	22518	6811
HBTree[35]	76.64	51.70	4.479	203.2	4764	945	**VinePlace**	67.41	10.90	**322.8**	**6.736**	**21946**	**5673**
VinePlace	**77.83**	**57.24**	**8.216**	**125.7**	**3753**	**845**	–	–	–	–	–	–	–
Ratio	–	–	–	–	**1.27×**	**1.07×**	**Ratio**	–	–	–	–	**1.03×**	**1.20×**

From the experimental results, it can be observed that topology-representation-based methods are more effective than gradient-based analytical placement methods in reducing layout area and wire length, leading to more compact analog layouts. In particular, the core-centered vine representation better organizes critical modules and helps preserve favorable post-layout performance. Meanwhile, the peripheral-around-core strategy together with geometry refinement improves geometric flexibility, resulting in a more compact physical implementation than conventional tree-based organizations. As a result, VinePlace achieves both superior circuit performance and strong layout compactness on this classical op-amp topology.

4.2.2. Comparator

For the comparator benchmark, VinePlace achieves the best overall results among the compared automated methods, especially on parasitic-sensitive metrics. It attains the lowest input-referred offset of 180.5 μV and the shortest delay of 0.939 ns. In addition, VinePlace produces the lowest output noise of 115.0 μV among all automated baselines. In physical design quality, VinePlace further reduces the layout area to 834 μm^2 and the HPWL to 77 μm, achieving 1.08× and 1.71× improvements over the best competing automated baseline, respectively. These results suggest that VinePlace is particularly effective for matching-critical comparator structures, where both device symmetry and routing parasitics strongly affect post-layout offset, delay, and noise.

4.2.3. Voltage-Controlled Oscillator (VCO)

On the VCO benchmark, VinePlace achieves the most compact layout among all automated methods while maintaining competitive oscillation characteristics. It obtains the highest oscillation frequency (2.85 GHz) and K_{vco} (1.59). VinePlace also reduces the area to 718 μm^2 and the HPWL to 282 μm, outperforming the best competing automated baseline by 1.21× and 1.07×, respectively. Despite not achieving the best phase noise, VinePlace provides a favorable balance between electrical performance and layout compactness.

4.2.4. Three-Stage OTA

For the three-stage OTA, VinePlace achieves the best post-layout performance among all compared methods, with the highest gain

(77.83 dB), phase margin (57.24°), and UGB (8.216 MHz), as well as the lowest noise (125.7 μV). Meanwhile, it produces the most compact layout, reducing the area to 3753 μm^2 and the HPWL to 845 μm, corresponding to 1.27× and 1.07× reductions, respectively, over the best-performing baselines. These results show that VinePlace can effectively reduce layout area and wirelength while preserving post-layout circuit performance, even for a multi-stage feedback circuit that is sensitive to routing parasitics.

4.2.5. SAR ADC

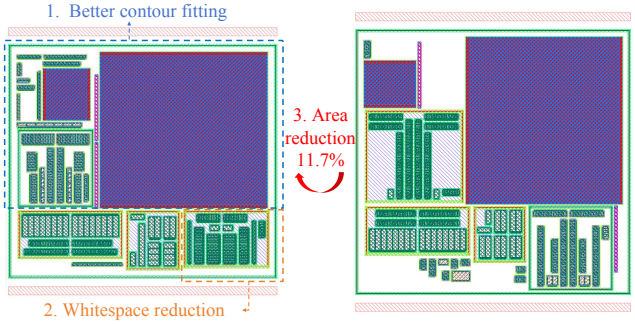
For the large-scale SAR ADC benchmark, VinePlace shows strong overall optimization capability on hierarchical mixed-signal layouts. Because this benchmark involves a hierarchical AMS structure, only JointPlace is applicable, whereas HBTree and MAGICAL do not support such circuits. Accordingly, we compare VinePlace with JointPlace and further include an expert-crafted human design as a practical reference. Although its SNDR and ENOB are slightly lower than those of JointPlace, the gaps remain marginal, with VinePlace achieving 67.41 dB SNDR and 10.90-bit ENOB. Meanwhile, VinePlace attains the lowest core power consumption of 322.8 μW and the best FOM of 6.736 fJ among the compared automated methods. In terms of physical design, VinePlace further reduces the area to 21946 μm^2 and the HPWL to 5673 μm, corresponding to 1.03× and 1.20× improvements over the best competing baseline.

More importantly, the layout generated by VinePlace approaches the quality of an expert-crafted human design, which typically requires weeks of manual effort. Its post-layout performance remains competitive, while both area and HPWL are also close to the human baseline. These results suggest that VinePlace can generate high-quality layouts in an automated flow, making it a promising solution for large hierarchical mixed-signal circuits such as SAR ADCs.

4.3. Ablation Study

Table 4 reports the ablation study on the three-stage OTA benchmark. We compare the full VinePlace with three reduced variants: *No Geo.*, which removes geometry-level refinement; *No Core*, which removes the core-centered organization; and *2-Dir Tree*, which further degenerates the method into a conventional two-direction tree-based organization.

Overall, the full VinePlace achieves the best trade-off between electrical performance and layout quality, obtaining the highest gain, the lowest noise, the smallest area, and the shortest HPWL.



(a) VinePlace with Geo. (b) VinePlace without Geo.

Figure 9: Effect of geometry refinement on placement.

Removing geometry-level refinement (*No Geo.*) noticeably worsens noise, area, and HPWL, showing its importance for local contour fitting and whitespace reduction. Removing the core-centered organization (*No Core*) causes a clear gain drop, indicating that core-centered placement helps preserve post-layout performance. Further reducing the method to a two-direction tree (*2-Dir Tree*) leads to the worst area, HPWL, and phase margin. As shown in Fig. 9, geometry-level refinement achieves better contour fitting, reduces internal whitespace, and further decreases the overall placement area by 11.7%. These results demonstrate that core-centered organization establishes a global structure that is beneficial for post-layout performance, while geometry-level refinement further improves local compactness, and both are important for compact and high-quality analog placement.

Table 4: Ablation results on the three-stage OTA benchmark.

Method	Gain (dB)	PM (°)	UGB (MHz)	Noise (μV)	Area (μm^2)	HPWL (μm)
No Geo.	77.15	**59.89**	4.469	158.4	4250	1018
No Core	73.16	55.48	7.586	153.1	3810	935
2-Dir Tree	74.96	43.01	4.627	140.1	4480	1193
Full	**77.83**	57.24	**8.216**	**125.7**	**3753**	**845**

4.4. Runtime Comparison

As shown in Table 5, VinePlace achieves the shortest runtime on all five benchmarks. On the largest SAR ADC, it completes optimization in 58.95 s, approximately 2× faster than JointPlace. This speedup arises from the nested vine representation, in which a perturbation requires repacking only the modified bottom-level vine and the top-level vine. The remaining unaffected bottom-level vines retain their packed results, avoiding redundant computation and reducing the packing overhead for hierarchical AMS circuits.

Table 5: Runtime (s) comparison on five circuits.

Method	Two stage	Comparator	VCO	Three stage	SAR ADC
Magical[40]	3.14	6.32	23.12	18.65	-
HBTree[35]	0.94	2.59	7.42	5.01	-
JointPlace[12]	2.97	4.58	11.66	12.29	118.54
VinePlace	**0.78**	**1.31**	**3.75**	**3.79**	**58.95**

5. Conclusion

In this paper, we proposed VinePlace, a core-centered placement framework for compact AMS layout generation. VinePlace combines a unified vine representation for diverse AMS structures with representation-aware perturbations and hierarchical packing. Experiments on five benchmarks, ranging from a 17-device two-stage OTA to a 1,623-device SAR ADC, demonstrate that VinePlace achieves competitive post-layout performance while reducing area by 1.03×–1.52× and HPWL by 1.07×–1.71× over the best-performing baselines. Results on the SAR ADC further demonstrate its scalability to large hierarchical mixed-signal circuits and its ability to generate high-quality layouts in an automated flow.

6. Acknowledgement

This project was supported in part by Beijing High Innovation Plan (202604841450) and Jiangsu National Science and Technology Major Project (SBG20250000204).

References

[1] Helmut Graeb, Florin Balasa, Rafael Castro-López, Y-W Chang, Francisco V Fernandez, P-H Lin, and Martin Strasser. Analog layout synthesis-recent advances in topological approaches. In *2009 Design, Automation & Test in Europe Conference & Exhibition*, pages 274–279. IEEE, 2009.

[2] Yiu-Cheong Tam, Evangeline FY Young, and Chris Chu. Analog placement with symmetry and other placement constraints. In *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, pages 349–354, 2006.

[3] Po-Hung Lin, Yao-Wen Chang, and Shyh-Chang Lin. Analog placement based on symmetry-island formulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(6):791–804, 2009.

[4] Linfu Xiao and Evangeline FY Young. Analog placement with common centroid and 1-d symmetry constraints. In *2009 Asia and South Pacific Design Automation Conference*, pages 353–360. IEEE, 2009.

[5] Po-Hung Lin, Hongbo Zhang, Martin DF Wong, and Yao-Wen Chang. Thermal-driven analog placement considering device matching. In *Proceedings of the 46th Annual Design Automation Conference*, pages 593–598, 2009.

[6] Qiang Ma, Evangeline F. Y. Young, and Kong-Pang Pun. Analog placement with common centroid constraints. In *Proceedings of the 2007 IEEE/ACM International Conference on Computer-Aided Design*, pages 579–585, 2007.

[7] Arvind K. Sharma, Meghna Madhusudan, Steven M. Burns, Soner Yaldiz, Parijat Mukherjee, Ramesh Harjani, and Sachin S. Sapatnekar. Performance-aware common-centroid placement and routing of transistor arrays in analog circuits. In *2021 IEEE/ACM International Conference on Computer Aided Design*, pages 1–9, 2021.

[8] Xiaohan Gao, Haoyi Zhang, Mingjie Liu, Linxiao Shen, David Z Pan, Yibo Lin, Runsheng Wang, and Ru Huang. Interactive analog layout editing with instant placement and routing legalization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(3):698–711, 2022.

[9] Hung-Chih Ou, Kai-Han Tseng, Jhao-Yan Liu, I-Peng Wu, and Yao-Wen Chang. Layout-dependent-effects-aware analytical analog placement. In *Proceedings of the 52nd Annual Design Automation Conference*, pages 1–6, 2015.

[10] Biying Xu, Shaolan Li, Chak-Wa Pui, Derong Liu, Linxiao Shen, Yibo Lin, Nan Sun, and David Z Pan. Device layer-aware analytical placement for analog circuits. In *Proceedings of the 2019 International Symposium on Physical Design*, pages 19–26, 2019.

[11] Keren Zhu, Hao Chen, Mingjie Liu, and David Z Pan. Hierarchical analog and mixed-signal circuit placement considering system signal flow. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(8):2689–2702, 2022.

[12] Xiaohan Gao, Haoyi Zhang, Bingyang Liu, Yibo Lin, Runsheng Wang, and Ru Huang. Joint placement optimization for hierarchical analog/mixed-signal circuits. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2024.

[13] Daniel Guerra, António Canelas, Ricardo Póvoa, Nuno Horta, Nuno Lourenço, and Ricardo Martins. Artificial neural networks as an alternative for automatic analog ic placement. In *2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pages 1–4. IEEE, 2019.

[14] Yaguang Li, Yishuang Lin, Meghna Madhusudan, Arvind Sharma, Wenbin Xu, Sachin Sapatnekar, Ramesh Harjani, and Jiang Hu. Exploring a machine learning approach to performance driven analog ic placement. In *2020 IEEE computer society annual symposium on VLSI (ISVLSI)*, pages 24–29. IEEE, 2020.

[15] Yaguang Li, Yishuang Lin, Meghna Madhusudan, Arvind Sharma, Wenbin Xu, Sachin S. Sapatnekar, Ramesh Harjani, and Jiang Hu. A customized graph neural network model for guiding analog ic placement. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 2020.

[16] Mingjie Liu, Keren Zhu, Jiaqi Gu, Linxiao Shen, Xiyuan Tang, Nan Sun, and David Z Pan. Towards decrypting the art of analog layout: Placement quality prediction via transfer learning. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020.

[17] Chen-Chia Chang, Jingyu Pan, Zhiyao Xie, Yaguang Li, Yishuang Lin, Jiang Hu, and Yiran Chen. Fully automated machine learning model development for analog placement quality prediction. In *Asia and South Pacific Design Automation Conference*, 2023.

[18] Ahmet F Budak, Keren Zhu, and David Z Pan. Practical layout-aware analog/mixed-signal design automation with bayesian neural networks. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–8. IEEE, 2023.

[19] Keren Zhu, Hao Chen, Mingjie Liu, Xiyuan Tang, Wei Shi, Nan Sun, and David Z. Pan. Generative-adversarial-network-guided well-aware placement for analog circuits. In *Asia and South Pacific Design Automation Conference*, pages 519–525, 2022.

[20] Lijie Wang, Jing Wang, Song Chen, and Qi Xu. Aiplace: Analog ic placement with multi-task learning framework. In *Proceedings of the 30th Asia and South Pacific Design Automation Conference*, pages 204–210, 2025.

[21] Xinfei Liu, Siting Liu, Bei Yu, Song Chen, and Qi Xu. Theplace: Thermal-aware placement with operator learning-based ultra-fast simulator. In *Proceedings of the 30th Asia and South Pacific Design Automation Conference*, pages 900–906, 2025.

[22] Keren Zhu, Hao Chen, Mingjie Liu, Xiyuan Tang, Nan Sun, and David Z. Pan. Effective analog/mixed-signal circuit placement considering system signal flow. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 2020.

[23] Tonmoy Dhar, S Ramprasath, Jitesh Poojary, Soner Yaldiz, Steven Burns, Ramesh Harjani, and Sachin S Sapatnekar. A charge flow formulation for guiding analog/mixed-signal placement. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 148–153. IEEE, 2022.

[24] Mingjie Liu, Keren Zhu, Xiyuan Tang, Biying Xu, Wei Shi, Nan Sun, and David Z Pan. Closing the design loop: Bayesian optimization assisted hierarchical analog layout synthesis. In *2020 57th ACM/IEEE design automation conference (DAC)*, pages 1–6. IEEE, 2020.

[25] Hongxia Zhou, Chiu-Wing Sham, and Hailong Yao. Revisiting routability-driven placement for analog and mixed-signal circuits. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 23(2):1–17, 2017.

[26] Hao Chen, Walker J Turner, David Z Pan, and Haoxing Ren. Routability-aware placement for advanced finfet mixed-signal circuits using satisfiability modulo theories. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 160–165. IEEE, 2022.

[27] S Ramprasath, Meghna Madhusudan, Arvind K Sharma, Jitesh Poojary, Soner Yaldiz, Ramesh Harjani, Steven M Burns, and Sachin S Sapatnekar. Analog/mixed-signal layout optimization using optimal well taps. In *ISPD*, pages 159–166, 2022.

[28] Arvind K Sharma, Meghna Madhusudan, Steven M Burns, Soner Yaldiz, Parijat Mukherjee, Ramesh Harjani, and Sachin S Sapatnekar. Constructive place-and-route for finfet-based transistor arrays in analog circuits under nonlinear gradients. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 43(12):4373–4385, 2024.

[29] Abhishek Patyal, Hung-Ming Chen, Mark Po-Hung Lin, Guan-Qi Fang, and Simon Yi-Hung Chen. Pole-aware analog layout synthesis considering monotonic current flows and wire crossings. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(1):266–279, 2022.

[30] Florin Balasa and Koen Lampaert. Symmetry within the sequence-pair representation in the context of placement for analog design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(7):721–731, 2002.

[31] Yingxin Pang, Florin Balasa, Koen Lampaert, and Chung-Kuan Cheng. Block placement with symmetry constraints based on the o-tree non-slicing representation. In *Proceedings of the 37th Annual Design Automation Conference*, pages 464–467, 2000.

[32] Florin Balasa and Sarat C Maruvada. Using non-slicing topological representations for analog placement. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 84(11):2785–2792, 2001.

[33] Jai-Ming Lin, Guang-Ming Wu, Yao-Wen Chang, and Jen-Hui Chuang. Placement with symmetry constraints for analog layout design using tcg-s. In *Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, pages 1135–1137, 2005.

[34] Lihong Zhang, C.-J. Richard Shi, and Yingtao Jiang. Symmetry-aware placement with transitive closure graphs for analog layout design. In *2008 Asia and South Pacific Design Automation Conference*, pages 180–185, 2008.

[35] Po-Hung Lin and Shyh-Chang Lin. Analog placement based on hierarchical module clustering. In *Proceedings of the 45th annual Design Automation Conference*, pages 50–55, 2008.

[36] Hui-Fang Tsao, Pang-Yen Chou, Shih-Lun Huang, Yao-Wen Chang, Mark Po-Hung Lin, Duan-Ping Chen, and Dick Liu. A corner stitching compliant b-tree representation and its applications to analog placement. In *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 507–511. IEEE, 2011.

[37] I-Peng Wu, Hung-Chih Ou, and Yao-Wen Chang. Qb-trees: Towards an optimal topological representation and its applications to analog layout designs. In *Proceedings of the 53rd Annual Design Automation Conference*, pages 1–6, 2016.

[38] Haoyi Zhang, Weijian Fan, Xiaohan Gao, Bingyang Liu, Runsheng Wang, and Yibo Lin. PANDA: An LLM-enhanced performance-driven analog design framework bridging design intent and layout generation. In *Proceedings of the 63rd ACM/IEEE Design Automation Conference (DAC)*, 2026.

[39] Haoyi Zhang, Xiaohan Gao, Haoyang Luo, Jiahao Song, Xiyuan Tang, Junhua Liu, Yibo Lin, Runsheng Wang, and Ru Huang. Sageroute: Synergistic analog routing considering geometric and electrical constraints with manual design compatibility. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2023.

[40] Biying Xu, Keren Zhu, Mingjie Liu, Yibo Lin, Shaolan Li, Xiyuan Tang, Nan Sun, and David Z Pan. Magical: Toward fully automated analog ic layout leveraging human and machine intelligence. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2019.