Joint Placement Optimization for Hierarchical Analog/Mixed-Signal Circuits

Xiaohan Gao[1,2,3], Haoyi Zhang[2,3], Bingyang Liu[4], Yibo Lin[2,3,5*], Runsheng Wang[2,3,5], Ru Huang[2,3,5]
{[1]School of Computer Science, [2]School of Integrated Circuits, [3]Institute of EDA, [4]School of EECS}, Peking University
[5]Beijing Advanced Innovation Center for Integrated Circuits
{xiaohangao, hy.zhang, 2100012715, yibolin, r.wang, ruhuang}@pku.edu.cn

Abstract

The performance of Analog/Mixed Signal (AMS) circuits is highly dependent on the meticulous layout implementation. To meet performance and area requirements, real-world AMS layout design is thoroughly optimized to consider circuit hierarchy and a multitude of factors, such as system signal flow and regularity. Circuit hierarchy and these factors impose complicated constraints, which challenge layout design flow. In this paper, we propose a systematic AMS placement framework to address the challenges through joint optimization. We implement our framework in a unified and highly extensible workflow and validate our framework with broad types of real-world AMS circuits. Experiments show that our framework achieves promising results in both efficiency and quality.

1 Introduction

AMS circuits have carved out a place in emerging applications, including the automotive industry, and Internet of Things [1]. The wide applications, in turn, call for a fast design closure with layout automation for AMS design. However, different AMS designs have to satisfy various demands for low noise, low delay, high gain, low area, and *etc.* To meet such demands, manual efforts often have to be involved for a holistic consideration of complex and compound factors. Indeed, those factors are one of the reasons why AMS layout still heavily relies on manual design even today.

Placement is not only the start but also one of the most dominant stages of AMS layout automation. The mainstream factors for AMS placement including (a) *Circuit hierarchy*: circuit hierarchy of AMS circuits typically requires a separate process, either in a bottom-up or a multi-level manner [2–4], which is not able to globally optimize the whole circuit layout; (b) *System signal flow/Power flow*: system signal flow (SSF) and power flow (SPF) introduce global signal patterns for AMS layout [5–8]; (c) *Regularity*: common regularity includes symmetry and collinear regularity [9, 10]; (d) *Area and Wirelength*: area and wirelength have been studied in many previous work [2, 3, 11, 12], but note that there is a trade-off between area and routability.

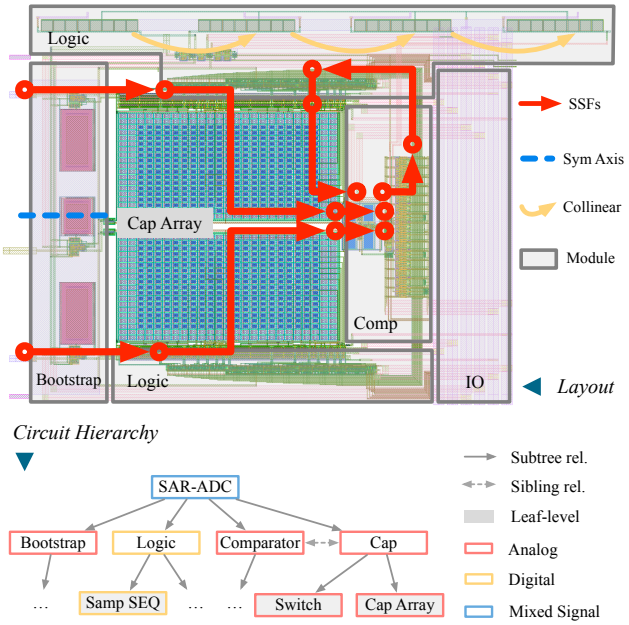

Figure 1: A real-world AMS design with circuit hierarchy.

The main challenges for AMS placement lie in two folds. 1) How to consider the circuit hierarchy? As AMS designers usually design circuits in a hierarchical manner, it is natural to emphasize the valuable hierarchy information at the placement stage. 2) How to holistically consider the aforementioned factors under versatile circuit structures? Figure 1 shows a real-world AMS circuit design that involves a complex hierarchy of circuit modules. The circuit design touches on various factors of placement, including system signal flows, symmetry, and collinear regularity. To fit in the sophisticated AMS design, a holistic and systematic placement framework is expected.

Previous efforts have explored formulating the versatile factors in AMS layout problems into a series of mathematical constraints and objectives to optimize [13, 14], such as considering symmetry and collinear regularity [9, 10]. However, the previous work fails to handle multiple factors under circuit hierarchy. To address the gap, we expect a systematic placement framework to model all aforementioned factors as unified optimization objectives. Even given objectives and constraints, how to optimize them is another big challenge. Digital placement engines usually consider no more than three objectives like

* Corresponding author.

ICCAD '24, October 27–31, 2024, New York, NY, USA

ACM ISBN 979-8-4007-1077-3/24/10
https://doi.org/10.1145/3676536.3676664

routability and timing at the same time, while AMS placement often needs to tackle more than five objectives simultaneously in one circuit. Irregular circuit hierarchy introduces additional complexity to the placement modeling and solving.

In this paper, we propose a systematic placement framework for AMS circuits, to jointly optimize complicated constraints and objectives. Our framework manages to consider circuit hierarchy simultaneously with placement constraints and objectives. The framework provides a unified and highly extensible way to accommodate various constraints and objectives. We summarize our main contributions as follows:

(1) We propose a systematic framework for AMS placement. The framework models the mainstream layout constraints and objectives in a unified way and jointly optimizes all the constraints and objectives;
(2) We model the circuit hierarchy as a convex objective and thus make it possible to thoroughly consider circuit hierarchy while optimizing other objectives;
(3) We propose a hierarchical legalization method generating a compact layout for high-performance placement;
(4) We evaluate our framework on different types of real-world AMS circuit designs. Experimental results show the ability of our framework to approximate the performance of tapeout layout.

We organize the remaining sections as follows. Section 2 introduces the background and our problem formulation. Section 3 briefs the workflow. Section 4 details the objectives and constraints we consider in our framework. Section 5 describes the algorithms of analytical placement, and Section 6 describes the algorithms of hierarchical legalization. Section 7 demonstrates the experimental results. Section 8 concludes this work.

2 Preliminaries

In this section, we review the prior work of hierarchical placement. We provide definitions of the placement objectives and give a problem formulation for hierarchical placement.

2.1 Hierarchical Placement

Standard AMS circuit design typically employs a hierarchical manner. The schematic design is equivalent to a tree hierarchy of modules (namely the sub-circuit blocks). Figure 1 shows an example hierarchy of modules extracted from a SAR-ADC circuit. Placement mainly considers the subtree relationship and the sibling relationship, which are the most valuable information in the circuit hierarchy.

Prior work with non-analytical placement addresses circuit hierarchy in a floorplan-like manner [8, 15–17]. More recent placement frameworks process circuits with hierarchy through a bottom-up procedure [2, 3]. By a bottom-up hierarchical procedure, the placer builds up the leaf-level modules and gradually generates layout for the upper hierarchical level. However, the bottom-up placement ignores the context in which a module is situated. In other words, when placing devices for a module, the placer disregards the upper-level circuit structure. The work [4] introduces an improved scheme to better plan system signal paths in lower-level modules, which requires an additional stage to modify the hierarchical layout. Such an approach still lacks sufficient flexibility in incorporating more constraints other than system signal paths and module shapes.

Table 1: Comparison to previous work.

Method	Hierarchy	Solver	Supported Objectives
LAYGEN II [18]	Bottom-up	S.A.	{Area, Sym}
MAGICAL [3, 4]	Multi-level	Nonlinear Opt.	{Area, WL, Sym, SSF}
ALIGN [2]	Bottom-up	S.A.	{Area, WL, Sym, Reg}
Ours	Simultaneous	Nonlinear Opt.	{Area, WL, Sym, SSF/SPF, Reg}

* S.A. for Simulated Annealing, Sym for Symmetry, WL for wirelength, SSF/SPF for system signal/power flow, Reg for Collinear Regularity.

Table 1 compares our framework with previous AMS layout tools. Our framework develops a nonlinear-optimization-based method that simultaneously considers circuit hierarchy and other placement objectives. Also, our framework is capable of optimizing most of the mainstream objectives.

2.2 Objectives and Constraints

To optimize performance and area, analytical placement formulates influential factors as objectives or constraints.

2.2.1 System Signal Flow and Power Flow. In AMS circuit, system signal flows define the paths through which the critical signals pass [6, 19, 20]. Power flow or charge flow defines the critical path of power networking [5]. Recent work proposes a cosine-like objective function for system signal flow [7], and the objective guides the modules in a monotonic direction along a system signal flow. In general cases, we observe that taking a Z shape or a semi-circular shape (like the SSFs in Figure 1) does not necessarily degrade performance. However, it is problematic if a system signal flow or power flow crosses over itself. To compensate for the cosine-like objective function, our framework additionally penalizes the crossing-over patterns.

2.2.2 Wirelength and Area. Routed wirelength is an important metric for AMS circuit. AMS automation tools consider half-perimeter wirelength or other approximation at the placement stage [21–23]. Area or boundary is another most commonly used objective [24, 25].

2.2.3 Symmetry and Collinear Regularity. Regularity-oriented placement considers symmetry and collinear regularity, which helps match the layout-induced parasitics between groups of devices [9, 10, 26].

2.2.4 Non-Overlap. Non-Overlap is the criterion that requires no spatial overlap between any two devices. This requirement can be formulated as a constraint between a pair of devices [27].

2.3 Constraint Graph and Legalization

Legalization is crucial to analytical optimization-based placement. An analytical placer based on nonlinear optimization does not guarantee a non-overlapping layout. The legalization first needs to determine the relative positions of the devices. A typical representation of relative positions is the constraint graph. Constraint graph consists of two directed acyclic graphs, namely horizontal constraint graph and vertical constraint

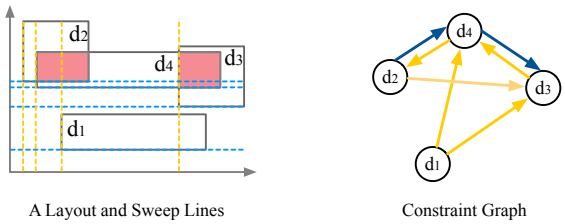

Figure 2: Sweep lines and constraint graph.

graph. The horizontal constraint graph contains horizontal constraint edges between two devices which represent the left-right relationship, and *vice versa*.

Figure 2 illustrates an example of 4 devices $\{d_1, d_2, d_3, d_4\}$. The sweep line algorithm is applied to compacting layout in one dimension on constraint graph [28]. We adopt the sweep line algorithm to construct constraint graph for legalization.

2.4 Problem Formulation

We consider a hierarchical AMS placement problem:

Problem 1 (Hierarchical Placement)**.** Given the schematic of a hierarchical AMS circuit, including the devices $\{d_i\}$, the circuit modules $\{C_i\}$, the pins $\{p_i\}$, and the nets $\{N_i\}$, determine the coordinates of all devices. If constraints of symmetry groups, collinear regularity groups, system signal flows, and power flows are specified, the determined coordinates need to meet the specifications. The objectives of wirelength, area, and constraints are globally optimized under the circuit hierarchy.

3 Placement Framework

Figure 3 demonstrates the overall workflow of our placement framework. In our framework, we consider circuit hierarchy simultaneously with other objectives and constraints, including signal flow and power flow, symmetry and collinear regularity, wirelength, area, and non-overlap. The hierarchical placement stage leverages the skeleton of analytical placement to initialize the objectives and solve the joint optimization. After placement, we apply hierarchical legalization based on constraint graph and linear programming to the placed results and produce legalized placement layouts. Our framework generates compact well regions to finish the final placement layouts. This paper further introduces the detailed algorithms in corresponding sections listed in Figure 3.

Table 2 defines the notation of symbols and objective functions we use in this paper. In Section 4, we give a detailed description of the objective functions and constraints for our analytical placement. In Section 5 and Section 6, we introduce the algorithms of placement and legalization which optimize those objectives and constraints and produce final layouts.

4 Modeling Objectives

In this section, we introduce the formulation of objectives and constraints considered in our hierarchical placement.

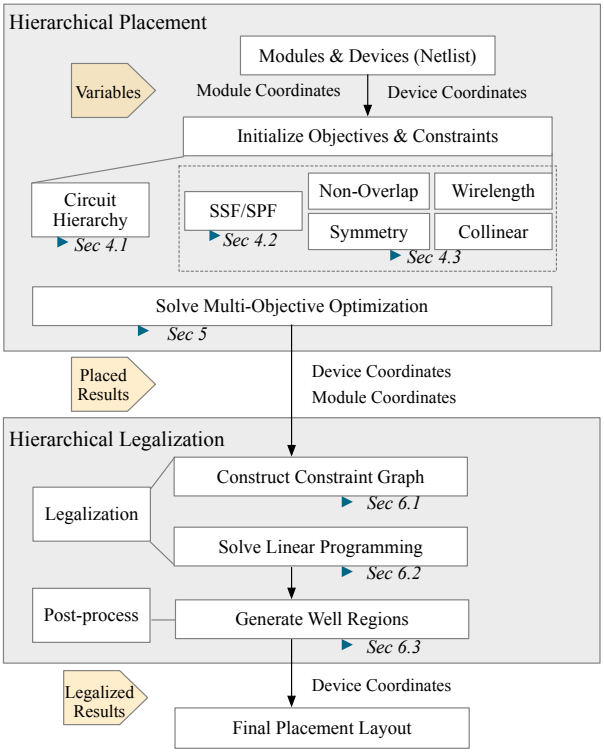

Figure 3: The overall workflow of our framework.

Table 2: Notation

Symbol	Description
d_i	The devices
p_i	The pins
(x_i, y_i)	The coordinate of corresponding device
(x_i^p, y_i^p)	The coordinate of corresponding pin
w_i, h_i	The width and height of corresponding device
C_j	The circuit modules
N_k	The nets
G	The corresponding hypergraph of circuit
(x_j^c, y_j^c)	The coordinate of corresponding sub-circuit block
A_j	The symmetry group
x^a, y^a	The symmetry axis of symmetry group
$\mathcal{P}_j^{SSF}, \mathcal{P}_j^{SPF}$	The system signal flows and the power flows
$\mathcal{R}_j^x, \mathcal{R}_j^y$	The collinear regularity groups
Objective	**Description**
$Obj^{Hier^I}, Obj^{Hier^O}$	The objective of circuit hierarchy
Obj^{SSF}, Obj^{SPF}	The objective of system signal flow or power flow
Obj^{LSEWL}	The objective of log-sum-exp wirelength
Obj^{OVL^I}, Obj^{OVL^O}	The constraint of non-overlap requirement
Obj^{Asym}, Obj^{Reg}	The constraint of symmetric and collinear regularity

4.1 Circuit Hierarchy as Objective

In order to consider circuit hierarchy while optimizing other objectives, we model the circuit hierarchy as a special objective. As we know from Figure 1, the key information of the circuit hierarchy hides in subtree relationships and sibling relationships. We model the two relationships separately with two objectives, defined as follows:

4.1.1 Intra-subtree Compactness. The modules of one subtree are supposed to stay closer to each other than to other modules outside the subtree. We define the intra-subtree compactness objective Obj^{Hier^I} under the module C_j as:

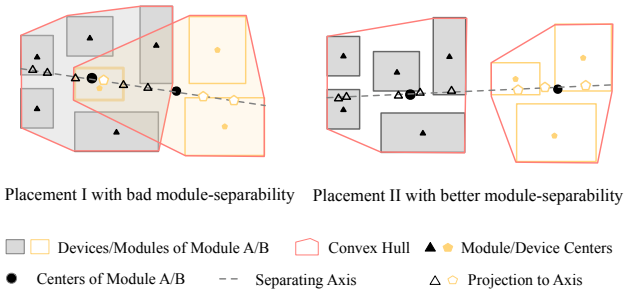

Figure 4: The example of inter-module separability.

$$Obj_{C_j}^{Hier^I} = \sum_{d_i \in C_j} ((x_i - x_j^c)^2 + (y_i - y_j^c)^2) + \sum_{C_k \in C_j} (x_k^c - x_j^c)^2 + (y_k^c - y_j^c)^2 \quad (1)$$

4.1.2 Inter-Module Separability. For sibling relationships, circuit hierarchy requires that a module be spatially separated from its sibling module. As shown in Figure 4, Placement I is worse than Placement II because one device of Module B is surrounded by devices of Module A. We expect that the convex hulls of sibling modules do not intersect with each other. In the collision detection domain, the *Separating Axis Theorem* provides a criterion for determining the intersection.The theorem states that two closed convex sets have no intersection if there exists a separating axis (a line) onto which the projections of the two sets are disjoint.

Inspired by the *Separating Axis Theorem*, we develop a separability objective. As shown in Figure 4, we approximately consider the line connecting the centers of two modules as a separating axis. The supposed separating axis is of the form $(y_A^c - y_B^c)(x - x_B^c) = (x_A^c - x_B^c)(y - y_B^c)$ for module A (x_A^c, y_A^c) and B (x_B^c, y_B^c). Let $a = y_A^c - y_B^c$, $b = x_B^c - x_A^c$, $c = x_A^c y_B^c - x_B^c y_A^c$. The projection of device center (x_i, y_j) onto the axis is $proj(x_i, y_i) = (\frac{b(-bx_i+ay_i)+ac}{-a^2-b^2}, \frac{bc-a(-bx_i+ay_i)}{-a^2-b^2})$. Calculate the projection of each device inside modules A and B to the separating axis. Projections of modules A and B cover two line segments whose overlap length defined the Obj^{Hier^O}:

$$\begin{aligned} Obj^{Hier^O} = &\min(\max_{d_i \in C_A} proj(x_i, y_i), \max_{d_j \in C_B} proj(x_j, y_j)) \\ &- \max(\min_{d_i \in C_A} proj(x_i, y_i), \min_{d_j \in C_B} proj(x_j, y_j)) \end{aligned} \quad (2)$$

The min and max formulation of Equation 2 is approximated by log-sum-exp function [29] in practice.

4.2 System Signal Flow and Power Flow

System signal flows and power flows are represented in a sequence of pin-to-pin connections.

Figure 5 shows two different placements I and II, but the cosine-lie objectives $4-(\cos\theta_1+\cos\theta_2+\cos\theta_3+\cos\theta_4)$ from [7] are quite close for the two placement. To compensate for the cosine-like objective, we propose a cross-product objective that emphasizes the penalty of the signal flow or power flow crossing itself. The right figure in Figure 5 demonstrates how to identify the crossing for placement II. A crossing occurs between the segment $p_0 \to p_1$ and $p_2 \to p_3$ when p_0, p_1 situate at different sides of $p_2 \to p_3$ and p_2, p_3 situates at different sides of $p_0 \to p_1$. That is to say, the cross product $(p_3 - p_0) \times (p_2 - p_0)$

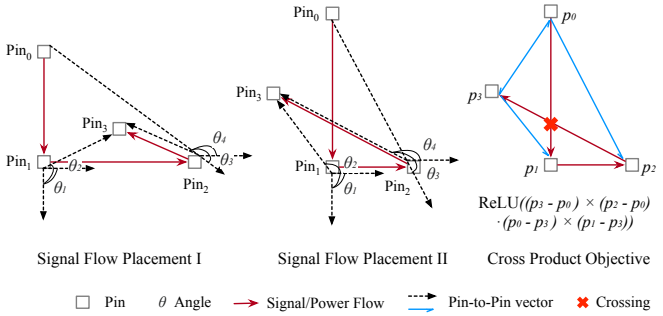

Figure 5: The example of signal flow objective.

has different sign of the cross product $(p_0 - p_3) \times (p_1 - p_3)$. Hence, we define the cross product objective Obj^{SSF} for signal flow (Obj^{SPF} for power flow shares the same formulation):

$$Obj^{SSF} = \sum_{\mathcal{P}_i^{SSF} \in \mathcal{P}^{SSF}} w_i^{SSF} f^{SSF}(\mathcal{P}_i^{SSF}) \quad (3a)$$

$$\begin{aligned} f^{SSF}(\mathcal{P}_i^{SSF}) = &\sum_{p_j \to p_{j+1} \in \mathcal{P}_i^{SSF}} \sum_{p_k \to p_{k+1} \in \mathcal{P}_i^{SSF}} \text{ReLU}(\\ &- ((p_k - p_j) \times (p_{k+1} - p_j)) \cdot ((p_j - p_k) \times (p_{j+1} - p_k))) \end{aligned} \quad (3b)$$

$$(p_k - p_j) \times (p_{k+1} - p_j) = (x_k^p - x_j^p)(y_{k+1}^p - y_j^p) - (x_{k+1}^p - x_j^p)(y_k^p - y_j^p) \quad (3c)$$

where the $\mathcal{P}_i^{SSF}$ is the i^{th} system signal flows, w_i^{SSF} is the weight factor for $\mathcal{P}_i^{SSF}$, $p_j \to p_{j+1}$ is the line segment connecting the adjacent pins in $\mathcal{P}_i^{SSF}$, $\text{ReLU}(x) = x^+$ is the rectified linear unit.

4.3 Generic Constraints

We consider other representative constraints, including wirelength, non-overlap, symmetry, and collinear regularity.

4.3.1 Wirelength. We adopt the log-sum-exp wirelength model for the wirelength objective Obj^{LSEWL} [27, 30].

$$\begin{aligned} Obj^{LSEWL} = \sum_{N_j} w_j^{WL} (&\log \sum_{d_i \in N_j} e^{x_i} + \log \sum_{d_i \in N_j} e^{-x_i} \\ &+ \log \sum_{d_i \in N_j} e^{y_i} + \log \sum_{d_i \in N_j} e^{-y_i}) \end{aligned} \quad (4)$$

4.3.2 Non-Overlap. Inside a leaf-level module, we adopts the one-to-one form for our intra-module overlap objective Obj^{OVL^I}:

$$\begin{aligned} Obj^{OVL^I} = \sum_{d_i} \sum_{d_j} &\max(min(x_i + w_i - x_j, x_j + w_j - x_i), 0) \\ &\cdot \max(min(y_i + h_i - y_j, y_j + h_j - y_i), 0) \end{aligned} \quad (5)$$

For the inter-module overlap, we adopts the module-to-module overlap objective Obj^{OVL^O} [27]:

$$\begin{aligned} Obj^{OVL^O} = \sum_{C_i} \sum_{C_j} &\max(min_{d_k \in C_i, d_m \in C_j}(x_k + w_k - x_m, x_m + w_m - x_k), 0) \\ &\cdot \max(min_{d_k \in C_i, d_m \in C_j}(y_k + h_k - y_m, y_m + h_m - y_k), 0) \end{aligned} \quad (6)$$

The min and max are approximated with log-sum-exp.

4.3.3 Symmetry. Symmetry constraints reduce the layout sensitivity for certain structures, such as differential pairs, current mirrors, or pairs of modules with the same circuit type.

We adopt the quadratic symmetry formulation [27]:

$$\begin{aligned} Obj^{Asym} = &\sum_{d_i \in A_j} (x_i - x_j^a)^2 + (y_i - y_j^a)^2 \\ &+ \sum_{C_i \in A_j} (x_i^c - x_j^a)^2 + (y_i^c - y_j^a)^2 \end{aligned} \quad (7)$$

4.3.4 Collinear Regularity. Collinear regularity requires modules to be arranged in a line or into rows. Collinear regularity could appear in digital parts or cascade structures of AMS designs. We propose the objective Obj^{Reg}:

$$\begin{aligned} Obj^{Reg} = &\sum_{\mathcal{R}_j^x} \sum_{C_i \in \mathcal{R}_j^x} |y_i^c - y_{i+1}^c| + |x_i^c - \hat{x}_i^c| \\ &+ \sum_{\mathcal{R}_j^y} \sum_{C_i \in \mathcal{R}_j^y} |x_i^c - x_{i+1}^c| + |y_i^c - \hat{x}_i^c| \\ \hat{x}_i^c &= x_l^c + (x_r^c - x_l^c)/|\mathcal{R}_j^x| \cdot (k-1),\ k^{th}\ module\ in\ \mathcal{R}_j^x \\ \hat{y}_j^c &= y_b^c + (y_u^c - y_b^c)/|\mathcal{R}_j^y| \cdot (k-1),\ k^{th}\ module\ in\ \mathcal{R}_j^y \end{aligned} \tag{8}$$

where for $\mathcal{R}^x$, x_l^c stands for the x coordinate of the first module, x_r^c for the last module, and for $\mathcal{R}^y$, y_b^c stands for the y coordinate of the first module, y_u^c for the last module.

5 Placement

In this section, we dive into the details of our placement algorithms. We introduce the analytical optimization formulation to optimize the objectives defined in Section 4.

5.1 Analytical Placement

We jointly optimize all the objectives proposed in Section 4. The placement problem is formulated as a multi-objective nonlinear optimization:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{x^c},\boldsymbol{y^c}} \quad & Obj^{WL}; Obj^{Hier}; Obj^{SSF} \\ s.t. \quad & Obj^{OVL^I} = 0, Obj^{OVL^O} = 0 \quad (Overlap) \\ & Obj^{Asym^I} = 0, Obj^{Asym^O} = 0 \quad (Symmetry) \\ & Obj^{Reg} = 0 \quad (Collinear) \end{aligned} \tag{9}$$

We relax the constraints in Equation 9 to the min objective:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{x^c},\boldsymbol{y^c}} \quad & f^{obj} = \lambda_{WL} \cdot Obj^{WL} + \lambda_{Hier} \cdot Obj_{Hier} \\ & + \lambda_{SSF} \cdot Obj^{SSF} + \lambda_{OVL} \cdot Obj^{OVL} \\ & + \lambda_{Asym} \cdot Obj^{Asym} + \lambda_{Reg} \cdot Obj^{Reg} \end{aligned} \tag{10}$$

where the $\lambda_{WL}, \lambda_{Hier}, \lambda_{SSF}, \lambda_{OVL}, \lambda_{Asym}, \lambda_{Array}$ denote multipliers for corresponding objectives.

5.2 Adaptive Moment Estimation

We adopt the adaptive moment estimation (Adam) operator [31, 32] to optimize f^{obj} of Equation 10. Algorithm 1 describes the iterative optimization process:

The algorithm first calculates gradient for all coordinate variables x_i, y_i and x_j^c, y_j^c with decay factor ξ (lines 3-4). Then we calculate the momentum $\hat{m}_t, \hat{v}_t$ with factor β_1 and β_2 (lines 5-7) and determine the descent direction δ_t (line 8). The gradient descent is applied to variables with step size γ (lines 9-11).

5.3 Multiplier and Step Size Update

To solve the placement fast, we update the multipliers and step size periodically.

5.3.1 λ update. In order to meet all the specifications for all objectives, Algorithm 1 dynamically adjust the multiplier λ (lines 12-13), including $\lambda_{Hier}, \lambda_{SSF}, \lambda_{OVL}, \lambda_{Reg}$ in Equation 10.

We initialize the multiplier λ with strategy from [7]:

Algorithm 1 Multi-objective Optimization

Input: Objective function f^{obj}, initial λ, and initial $\{(x_i, y_i)\}$
Output: The coordinates (x_i, y_i) of all devices d_i

1: **function** Adam(δ)
2: **while** not convergence **do**
3: $g_t \leftarrow \nabla_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{x^c},\boldsymbol{y^c}} f^{obj}(\boldsymbol{x},\boldsymbol{y},\boldsymbol{x^c},\boldsymbol{y^c})$
4: $g_t \leftarrow g_t + \xi(\boldsymbol{x},\boldsymbol{y},\boldsymbol{x^c},\boldsymbol{y^c})$
5: $\hat{m}_t \leftarrow (\beta_1 m_{t-1} + (1-\beta_1) g_t)/(1-\beta_1^t)$
6: $\hat{v}_t \leftarrow (\beta_2 v_{t-1} + (1-\beta_2) g_t^2)/(1-\beta_2^t)$
7: $\hat{v}_t^{max} \leftarrow \max(\hat{v}_t^{max}, \hat{v}_t)$
8: $\delta_t \leftarrow -\hat{m}_t/(\sqrt{\hat{v}_t^{max}} + \epsilon)$; $\boldsymbol{X} \leftarrow (\boldsymbol{x},\boldsymbol{y},\boldsymbol{x^c},\boldsymbol{y^c})$
9: **while** $f^{obj}(\boldsymbol{X} + \gamma\delta_t) > f^{obj}(\boldsymbol{X}) - c_1\gamma\delta_t^T\delta_t$ **do**
10: $\gamma \leftarrow c_2 \cdot \gamma$
11: $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{x^c},\boldsymbol{y^c}) \leftarrow (\boldsymbol{x},\boldsymbol{y},\boldsymbol{x^c},\boldsymbol{y^c}) + \gamma\delta_t$
12: **if** not meeting specifications **then**
13: update λ
14: **return** $\{(x_i, y_i)\}$

$$\lambda^0 = \|\nabla Obj^{WL}\|/\|\nabla Obj^s\|, s \in \{WL, Hier, SSF, OVL, Reg\} \tag{11}$$

We apply the subgradient update strategy for λ update:

$$\begin{aligned} \nabla_{sub}\lambda &= (\cdots, Obj_s + \frac{c_s}{2} Obj_s^2, \cdots)^T, s \in \{Hier, SSF, OVL, Reg\} \\ \lambda^{k+1} &= \lambda^{k+1} + t^k \nabla_{sub}\lambda^k / \|\nabla_{sub}\lambda^k\|_2 \end{aligned} \tag{12}$$

where c_s is a smooth constant, t^k is the step size for subgradient descent, $\|\cdot\|_2$ is the ℓ_2 norm.

5.3.2 Step Size Update. To achieve fast convergence for Algorithm 1, we apply line search technique for step size update. We iteratively find the largest step size γ that satisfies the Armijo condition (lines 9-10). The Armijo condition $f^{obj}(\boldsymbol{X} + \gamma\delta_t) > f^{obj}(\boldsymbol{X}) - c_1\gamma\delta_t^T\delta_t$ states that the new objective value after taking a descent step should be less than the current objective value plus a scaled decrease. The Armijo condition helps ensure that the step size leads to a sufficient decrease in objective function. c_1 and c_2 are two constants and we set them to 0.85 and 0.5 for experiments.

6 Legalization

Analytical placement produces a rough solution of device positions and the subsequent legalization stage is required to eliminate the remaining illegal overlaps between devices. We propose a hierarchical legalization method derived from the constraint graph-based legalization. Our legalization method first determines the relative positions of the devices and performs linear programming on the relative positions to get the legalized positions. To complete the layout, we generate well regions for MOSFET devices.

6.1 Constraint Graph Construction

We adopt the constraint graph representation for our legalization process. We initialize the constraint graph by the sweep line algorithm described in Section 2.3.

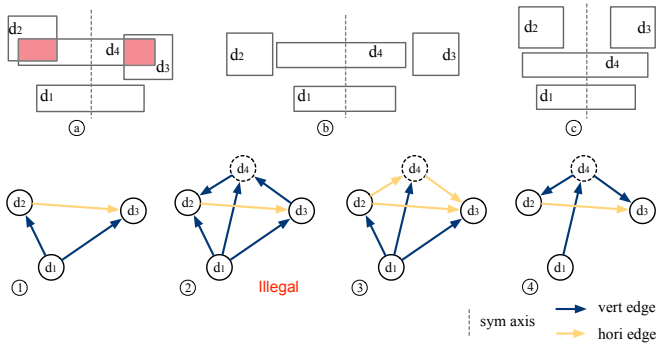

Figure 6: Constraint graph considering symmetry group $\{\{d_1\}, \{d_2, d_3\}\}$. ① Constraint graph among symmetry group; ② An illegal constraint graph; ③, ④ Legal constraint graphs with corresponding layouts ⓑ and ⓒ.

However, the initial constraint graph cannot consider the placement constraints including symmetry regularity, or ensure a compact layout. The first problem is that the constraint graph could have multiple redundant edges. For example, if there is a path from device d_i to d_j in the horizontal constraint graph, the vertical constraint edge $\langle d_i, d_j \rangle$ could be redundant, and *vice versa*. The second problem is that the constraint graph could conflict with symmetry and collinear regularity. As illustrated in Figure 6②, directly applying the sweep line algorithm will result in an infeasible constraint graph.

To solve the two aforementioned problems, we first predefine constraint edges for regularity constraints. After applying the sweep line algorithm, we remove the redundant edges from the constraint graph. Then we complete the graph to ensure that there is a path between any two devices either in the horizontal constraint graph or in the vertical constraint graph.

6.1.1 Predefined Constraint Edges. To simultaneously satisfy the placement constraints and constraint graph, we pre-define constraint edges for symmetry and collinear regularity.

Figure 6 shows an example of legalization with symmetry constraints. The symmetry group contains a self symmetry $\{d_1\}$ and symmetry pair $\{d_2, d_3\}$. If we follow the conventional sweep line algorithm [27], we will get the constraint graph shown in Figure 6②. The path from device d_3 to d_2 ($d_3 \rightarrow d_4 \rightarrow d_2$) means device d_2 is supposed to be placed above d_3, which contradicts the requirement of symmetry constraint.

A legal constraint graph must be free of contradiction with regularity constraints. Our legalization constructs the constraint graph among the devices of the regularity group in advance. For example, the constraint graph of devices d_1, d_2, d_3 is shown in Figure 6①. When considering the device d_4, our legalization checks the potential conflicts by detecting whether adding an edge will induce a cycle in the constraint graph. Figure 6③ and ④ are two feasible solutions, and Figure 6ⓑ and ⓒ illustrate their corresponding legalized layouts. In practice, our legalization method only considers the solution of a constraint graph with minimum displacement.

6.1.2 Complete Constraint Graph. A complete constraint graph implies there is either a path in horizontal constraint edge or a path in vertical constraint graph between any two devices. To avoid unnecessary displacement, we remove redundant edges if the edge occurs in both horizontal and vertical constraint graphs. Therefore the generated constraint graph is highly likely to be incomplete. The primitive algorithm to complete the constraint graph is to traverse all pairs of two devices and complement an edge between them either in horizontal or vertical constraint graph. However, the primitive algorithm still introduces redundancy to constraint graph.

We propose a topological sort-based method to complete the constraint graph with less redundancy. If there is no path between device d_i and d_j, then we define the $\langle d_i, d_j \rangle$ and $\langle d_j, d_i \rangle$ as missing edges. We arrange all the missing edges by a certain partial order $\prec$. Let $\langle d_i, d_j \rangle$ and $\langle d_k, d_t \rangle$ be two missing edges, and $\langle d_i, d_j \rangle \prec \langle d_k, d_t \rangle$ denotes that given $\langle d_i, d_j \rangle$, $\langle d_k, d_t \rangle$ will not be a missing edge. We perform a topological sort on the missing edges by the partial order and give a sorting of the missing edges. We iterate the sorting and update the missing edges if some missing edge is eliminated by adding another missing edge to the constraint graph. This process finally constructs a constraint graph with few redundant edges for a better legalization performance.

6.2 Hierarchical Legalization

After we determine the relative positions, we perform linear programming to calculate the final positions for the devices. The objectives of linear programming can be minimizing the displacement or minimizing the total width and length.

6.2.1 Linear Programming Formulation. Here we present the linear programming formulation that minimizes the displacement and forces the layout to satisfy all constraints including symmetry and collinear regularity.

Let x'_i, y'_i be the original coordinates and x_i, y_i be the legalized coordinates. The linear programming formulation to minimize the displacement of the legalization is as follows:

$$
\begin{aligned}
\min_{x_i, y_i} \quad & \sum_i |x_i - x'_i| + \sum_i |y_i - y'_i| \\
s.t. \quad & x_i + x_j = 2x^a_k, y_i = y_j, d_i, d_j \in A^x_k && (Symmetry) \\
& y_i + y_j = 2y^a_k, x_i = x_j, d_i, d_j \in A^y_k && (Symmetry) \\
& x_i - x_j = c, d_i, d_j \in \mathcal{R}^x_k; y_i - y_j = c, d_i, d_j \in \mathcal{R}^y_k && (Collinear) \\
& x_i + w_i \le x_j, \langle d_i, d_j \rangle \in \mathcal{G}_{\mathcal{H}} && (Non-Overlap) \\
& y_i + h_i \le y_j, \langle d_i, d_j \rangle \in \mathcal{G}_{\mathcal{V}} && (Non-Overlap)
\end{aligned} \tag{13}
$$

where $\mathcal{G}_{\mathcal{H}}$ stands for the horizontal constraint graph and $\mathcal{G}_{\mathcal{V}}$ for the vertical one. By solving the Equation 13, we generate the legalized coordinates of each device for a sub-circuit block.

6.2.2 Linear Programming on Hierarchical Constraint Graph. For hierarchical circuit, we have two schemes for hierarchical legalization. The intuitive legalization method could apply a bottom-up legalization scheme. We arrange the hierarchical legalization of sub-circuit modules in the reverse order of Breath-First-Search. The legalized sub-circuit module will be regarded as a single block in the subsequent high-level legalization process.

Another scheme is to solve the legalization of the whole circuit simultaneously. For each sub-circuit block, we add 4 vertices $v^{xl}, v^{xh}, v^{yl}, v^{yh}$ to the constraint graph to represent

Algorithm 2 Cluster PMOS Devices for Well Generation

Input: PMOS devices V^{PMOS} and all devices V^{ALL}
Output: Clusters $Set_V = \{V = \{d_i\}\}$ of MOSFET devices
1: **function** `ConnectGraph`($V^{PMOS} = \{d_i\}, V^{ALL} = \{d_i\}$)
2: **for all** $d_i \in V^{PMOS}$ **do**
3: **for all** $d_j \in V^{PMOS}$ **do**
4: $Box \leftarrow$ `BoundBox`(d_i, d_j)
5: **if** $Box \cap (V^{ALL} - V^{PMOS}) = \emptyset$ **then**
6: $E^{PMOS} \leftarrow E^{PMOS} \cup \{(d_i, d_j)\}$
7: **return** $G^{PMOS} = (V^{PMOS}, E^{PMOS})$
8: **while** $V^{PMOS} \neq \emptyset$ **do**
9: $G^{PMOS} \leftarrow$ `ConnectGraph`(V^{PMOS}, V^{ALL})
10: $V^{Clique} \leftarrow$ `FindClique`(G^{PMOS})
11: $Set_V \leftarrow Set_V \cup V^{Clique}$
12: $V^{PMOS} \leftarrow V^{PMOS} - V^{Clique}$
13: **return** Set_V

4 sides of the block boundary. Inside a sub-circuit block, we add horizontal edges from v_{xl} to each device d_i, and the other 3 sides are similar. Between two sub-circuit blocks C_i and C_j at the same hierarchy depth, we either add a horizontal edge between v_i^{xh} and v_j^{xl} or add a vertical edge between v_i^{yh} and v_j^{yl}. Therefore we flatten the hierarchical legalization and solve the Equation 13 for all devices.

6.3 Well Generation

For following routing and manufacturing stage, we are supposed to define the well regions for MOSFET devices [33–37]. In AMS circuits, there are no predefined well regions like those in standard cell-based digital circuits. For example, N-well region serves as the BULK of p-channel MOSFET (PMOS) devices, and designers draw the N-well regions and N-well guard rings to enclose the PMOS devices separately.

For the compactness and routability of placement layouts, it is practical to share well regions between MOSFET devices. In an optimized placement, we divide PMOS devices into feasible clusters and generate a shared well region inside each cluster.

6.3.1 Cluster MOSFETs for Well Generation. We present our clustering algorithm in Algorithm 2 taking PMOS and N-well for example. For compactness, our clustering algorithm targets grouping neighbor PMOS devices as many as possible. We construct an undirected graph in which the graph edge represents the possibility of two devices appearing in the same cluster (lines 1-7). If there are no devices other than PMOS inside the bounding box of two PMOS devices, the algorithm assigns an edge between the two devices. We apply the Bron Kerbosch algorithm to find a clique (line 10). A clique of PMOS devices indicates that any two devices of the clique can share the same N-well region. A clique composes a PMOS cluster (line 11). We remove the generated cluster from the PMOS set (line 12) and recursively generate the cluster until every PMOS device belongs to a cluster (lines 8-9). Then we complete the N-well generation after well generation. Our framework encloses each cluster with N-well region and N-well guard ring.

7 Experimental Results

We implement our placement framework in C++ with LibTorch and GUROBI. The placement experiments are conducted on a Linux server with Intel Xeon Gold 6230 CPU @ 2.10GHz. We perform constraint detection before our placement using tools [38], and perform routing after our placement using an A-star-based analog router [39], in order to get the final layouts for post-layout simulation. The post-layout simulation is based on Cadence Spectre, Ultra APS, and Calibre PEX. We apply Calibre PEX to extract parasitic resistance, parasitic capacitance, and coupling capacitance (R+C+CC option).

7.1 Benchmarks and Baselines

7.1.1 ***Benchmarks.*** We summarize the statistics of our circuit benchmarks in Table 3. The benchmarks are from mainstream technologies of TSMC 28nm, TSMC 40nm, and TSMC 65nm. The benchmarks involve all the aforementioned constraints for placement, including circuit hierarchy, symmetry and collinear regularity, and system signal/power flow. To be specific, the SAR-ADC case is a tapeout design with more than 1500 devices, and optimizing SAR-ADC simultaneously involves circuit hierarchy, symmetry regularity, and system signal flow.

The schematic of OTA is obtained from the work [3] and we build a testbench for this case. The LDO and SAR-ADC are obtained from the work [39], and the CCO is obtained from a tapeout case [40] designed by an expert AMS designer.

7.1.2 ***Baselines.*** We compare our framework (denoted as Ours) with 3 baseline methods: (1) the analog placer from the open-source layout generator MAGICAL [3] (denoted as MAGICAL); (2) our placer without considering circuit hierarchy (denoted as Ours-w/o-hier); (3) the tapeout layouts (if there exists) with both placement and routing optimized by expert designer manually (denoted as Manual). In addition, we also provide the results of the schematic simulation (denoted as Schematic) as a reference.

We employ MAGICAL as our baseline for benchmarks OTA and LDO because MAGICAL is only compatible with those circuits of TSMC 40nm technology. For benchmarks with circuit hierarchy, including LDO, CCO, and SAR-ADC, we compare our results with Ours-w/o-hier. Ours-w/o-hier performs placement in a bottom-up manner to gradually place a hierarchical circuit. For CCO and SAR-ADC, which are published at top design venues [40], we compare with the tapeout results with manually placed and routed layouts.

We further demonstrate the circuits and their performance in the following subsections.

7.2 Performance and Comparison

7.2.1 ***OTA.*** OTA stands for an operational transconductance amplifier circuit. The schematic and post-layout performances of OTA are shown in Table 4. OTA is an open-source case from MAGICAL [3] without circuit hierarchy. For this case, we achieve similar performance with the source provider MAGICAL and better performance in UGB.

Table 3: Benchmark Statistics.

Benchmark	Technology	#Devices	#Nets	Die Size	Cir. Hierarchy	Symmetry Reg.	Collinear Reg.	Sys. Signal Flow	Sys. Power Flow
OTA	TSMC40	49	35	$67.4 \times 75.7 \mu m^2$	×	✓	×	×	×
LDO	TSMC40	13	13	$53.3 \times 71.7 \mu m^2$	✓	✓	×	×	×
CCO	TSMC28	84	33	$56.5 \times 10.1 \mu m^2$	✓	✓	✓	×	✓
SAR-ADC	TSMC65	1623	709	$240.6 \times 192.7 \mu m^2$	✓	✓	×	✓	×

Table 4: OTA Performance.

Method	Gain (dB)	UGB (MHz)	CMRR (dB)	PM (degree)	Runtime (s)
Schematic	38.63	6.85	-	70.98	-
MAGICAL [3]	**38.44**	5.10	**55.70**	70.43	44.21
Ours	38.34	**5.18**	53.47	69.70	11.24

Table 5: LDO Performance.

Method	Gain (dB)	Current (uA)	VOD (mV)	VOU (mV)	PM (degree)	Runtime (s)
Schematic	73.69	16.82	539.6	540.2	89.69	-
MAGICAL [3]	73.06	16.18	1937.0	1422.0	89.61	2.76
Ours-w/o-hier	73.16	16.18	1938.0	1424.0	89.58	2.59
Ours	**73.33**	**16.39**	**1281**	**844**	89.63	1.63

Table 6: CCO Performance.

Method	Power (nW)	Frequency (MHz)	Runtime (s)
Schematic	324.2	16.70	-
Manual	324.1	**8.27**	-
Ours-w/o-hier	**324.0**	7.29	11.27
Ours	**324.0**	7.41	13.78

7.2.2 ***LDO.*** LDO stands for a low-dropout regulator circuit. The LDO circuit prefers higher values in Gain and Current and lower values in overshot down voltage (denoted as VOD) and overshot up voltage (denoted as VOU), and requires the phase margin (denoted as PM) in a reasonable range (≤ 90 for this case). The schematic and post-layout performances are shown in Table 5. As a method without hierarchy consideration, Ours-w/o-hier achieves slightly better performance than MAGICAL. Meanwhile, Ours drastically outperforms the 2 baselines in all metrics of Gain, Current, VOD, and VOU, which are much closer to the requirements of the schematic simulation.

7.2.3 ***CCO.*** CCO stands for a current-controlled oscillator. Circuit hierarchy, power flow, collinear regularity, and symmetry are considered in the CCO design. A CCO circuit usually consists of multiple collinear sub-circuits to better satisfy the system power flow. The placement of CCO is supposed to take care of the circuit hierarchy along with its complicated regularity and power flow. The schematic and post-layout performances are shown in Table 6. Ours can achieve better power consumption and close performance in Frequency compared to the Manual result of manually placed and routed tapeout layout. Note that for this case, as the schematic simulation is not able to consider parasitic capacitance, there is a significant difference in Frequency metric between the schematic and post-layout simulation for the CCO case.

7.2.4 ***SAR-ADC.*** SAR-ADC stands for a successive approximation analog-to-digital converter. We place the whole circuit considering circuit hierarchy, complex signal flows, and symmetry regularity. For convenience, we extract the layout except for the capacitor array part and perform the simulation with the capacitor array in schematic domain. The schematic and post-layout performances are shown in Table 7. SINAD represents the signal-to-noise and distortion ratio, ENOB represents the effective number of bits, which are the higher the better. FoM represents the power consumption per conversion and Pcore for the core power. Delay, Pcore, and FoM are the lower the better. Ours achieves better performance of all metrics than Ours-w/o-hier. Compared to Manual of a meticulously designed layout by expert, Ours achieves better Delay and Pcore and close performance in other metrics.

Table 7: SAR-ADC Performance.

Method	Delay (ns)	SINAD (dB)	ENOB (bit)	Pcore (uW)	FoM (fJ/conv)	Runtime (s)
Schematic	20.34	66.56	10.76	203.4	4.684	-
Manual	28.10	**66.31**	**10.72**	257.0	**6.085**	-
Ours-w/o-hier	23.57	58.73	9.46	260.5	14.77	141.71
Ours	**22.85**	60.17	9.70	**255.2**	12.26	133.54

For the first three benchmarks, our framework generates placement in less than 20 seconds. For the SAR-ADC with more than 1500 devices, we can finish layout generation within three minutes. Our framework significantly outperforms the method considering circuit hierarchy in a bottom-up manner. Also, we produce placement layouts comparable to expert-drawn manual layouts in a relatively short time.

8 Conclusions

In this paper, we propose a systematic placement framework for hierarchical AMS circuits. We consider mainstream factors as objectives or constraints for AMS layout design, including symmetry and collinear regularity, system signal flow and power flow, wirelength, and area. We also model the circuit hierarchy as an objective and jointly optimize all those objectives. We develop efficient optimizers for hierarchical placement and hierarchical legalization. Our proposed framework is implemented in a highly extensible workflow and compatible with additional constraints or objectives. Experimental results on real-world complex design demonstrate the potential of achieving competitive performance compared with manually drawn layouts by expert designers.

acknowledgement

This work was supported in part by the National Science Foundation of China (Grant No. 62141404, 62125401), the Natural Science Foundation of Beijing, China (Grant No. Z230002), and the 111 project (B18001).

References

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

[2] K. Kunal, M. Madhusudan, A. K. Sharma *et al.*, "Invited: Align – open-source analog layout automation from the ground up," in *Proc. DAC*, 2019, pp. 1–4.

[3] B. Xu, K. Zhu, M. Liu *et al.*, "Magical: Toward fully automated analog ic layout leveraging human and machine intelligence: Invited paper," in *Proc. ICCAD*, 2019, pp. 1–8.

[4] K. Zhu, H. Chen, M. Liu *et al.*, "Hierarchical analog and mixed-signal circuit placement considering system signal flow," *IEEE TCAD*, pp. 1–1, 2022.

[5] T. Dhar, R. S, J. Poojary *et al.*, "A charge flow formulation for guiding analog/mixed-signal placement," in *Proc. DATE*, 2022, pp. 148–153.

[6] P.-H. Wu, M. P.-H. Lin, Y.-R. Chen *et al.*, "Performance-driven analog placement considering monotonic current paths," in *Proc. ICCAD*, 2012, pp. 613–619.

[7] K. Zhu, H. Chen, M. Liu *et al.*, "Effective analog/mixed-signal circuit placement considering system signal flow," in *Proc. ICCAD*, 2020, pp. 1–9.

[8] M. P.-H. Lin, P.-H. Chang, S.-Y. Lee *et al.*, "Demixgen: Deterministic mixed-signal layout generation with separated analog and digital signal paths," *IEEE TCAD*, vol. 35, no. 8, 2016.

[9] S. Nakatake, M. Kawakita, T. Ito *et al.*, "Regularity-oriented analog placement with diffusion sharing and well island generation," in *Proc. ASPDAC*, 2010, pp. 305–311.

[10] P.-Y. Chou, H.-C. Ou, and Y.-W. Chang, "Heterogeneous b*-trees for analog placement with symmetry and regularity considerations," in *Proc. ICCAD*, 2011, pp. 512–516.

[11] J. Cohn, D. Garrod, R. Rutenbar *et al.*, "Koan/anagram ii: new tools for device-level analog placement and routing," *IEEE Journal Solid-State Circuits*, vol. 26, no. 3, pp. 330–342, 1991.

[12] Y.-S. Lu, Y.-H. Chang, and Y.-W. Chang, "Wb-trees: A meshed tree representation for finfet analog layout designs," in *Proc. DAC*, 2018, pp. 1–6.

[13] R. A. Rutenbar, "Analog circuit and layout synthesis revisited," in *Proc. ISPD*, 2015.

[14] Y. Lin, Y. Li, D. Fang *et al.*, "Are analytical techniques worthwhile for analog ic placement?" in *Proc. DATE*, 2022, pp. 154–159.

[15] P.-H. Lin and S.-C. Lin, "Analog placement based on hierarchical module clustering," in *Proc. DAC*, 2008, pp. 50–55.

[16] B. Xu, S. Li, X. Xu *et al.*, "Hierarchical and analytical placement techniques for high-performance analog circuits," in *Proc. ISPD*, 2017, p. 55–62.

[17] Y. Li, Y. Lin, M. Madhusudan *et al.*, "A customized graph neural network model for guiding analog ic placement," in *Proc. ICCAD*, 2020, pp. 1–9.

[18] R. Martins, N. Lourenço, and N. Horta, "Laygen II: Automatic analog ics layout generator based on a template approach," in *Proc. GECCO*, 2012, p. 1127–1134.

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

[20] A. Patyal, P.-C. Pan, A. K.A. *et al.*, "Analog placement with current flow and symmetry constraints using pcp-sp," in *Proc. DAC*, 2018, pp. 1–6.

[21] J. Rijmenants, J. Litsios, T. Schwarz *et al.*, "Ilac: an automated layout tool for analog cmos circuits," *IEEE Journal Solid-State Circuits*, vol. 24, no. 2, pp. 417–425, 1989.

[22] C. Lin, C. Lu, J. Lin *et al.*, "Routability-driven placement algorithm for analog integrated circuits," in *Proc. ISPD*, 2012, pp. 71–78.

[23] H. Chen, W. J. Turner, D. Z. Pan *et al.*, "Routability-aware placement for advanced finfet mixed-signal circuits using satisfiability modulo theories," in *Proc. DATE*, 2022, pp. 160–165.

[24] C.-W. Lin, J.-M. Lin, C.-P. Huang *et al.*, "Performance-driven analog placement considering boundary constraint," in *Proc. DAC*, 2010, pp. 292–297.

[25] Q. Ma, L. Xiao, Y.-C. Tam *et al.*, "Simultaneous handling of symmetry, common centroid, and general placement constraints," *IEEE TCAD*, vol. 30, no. 1, pp. 85–95, 2011.

[26] F. Balasa, S. Maruvada, and K. Krishnamoorthy, "Efficient solution space exploration based on segment trees in analog placement with symmetry constraints," in *Proc. ICCAD*, 2002, pp. 497–502.

[27] B. Xu, S. Li, C.-W. Pui *et al.*, "Device layer-aware analytical placement for analog circuits," in *Proc. ISPD*, 2019.

[28] J. Doenhardt and T. Lengauer, "Algorithmic aspects of one-dimensional layout compaction," *IEEE TCAD*, vol. 6, no. 5, pp. 863–878, 1987.

[29] F. Nielsen and K. Sun, "Guaranteed bounds on information-theoretic measures of univariate mixtures using piecewise log-sum-exp inequalities," *Entropy*, vol. 18, no. 12, 2016.

[30] W. C. Naylor, R. Donelly, and L. Sha, "Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer," 2001, US Patent 6,301,693.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, Y. Bengio and Y. LeCun, Eds., 2015.

[32] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *Proc. ICLR*, 2018.

[33] B. Xu, Y. Lin, X. Tang *et al.*, "Wellgan: Generative-adversarial-network-guided well generation for analog/mixed-signal circuit layout," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, 2019, pp. 1–6.

[34] R. S. Gopalakrishnan, M. Madhusudan, A. K. Sharma *et al.*, "A generalized methodology for well island generation and well-tap insertion in analog/mixed-signal layouts," *ACM TODAES*, 2023.

[35] R. Martins, N. Lourenço, R. Póvoa *et al.*, "On the exploration of design tradeoffs in analog ic placement with layout-dependent effects," in *Proc. SMACD*, 2019, pp. 25–28.

[36] H.-C. Ou, K.-H. Tseng, J.-Y. Liu *et al.*, "Layout-dependent effects-aware analytical analog placement," *IEEE TCAD*, vol. 35, no. 8, pp. 1243–1254, 2016.

[37] A. K. Sharma, M. Madhusudan, S. M. Burns *et al.*, "Performance-aware common-centroid placement and routing of transistor arrays in analog circuits," in *Proc. ICCAD*, 2021, pp. 1–9.

[38] M. Liu, W. Li, K. Zhu *et al.*, "S3det: Detecting system symmetry constraints for analog circuits with graph similarity," in *Proc. ASPDAC*, 2020, pp. 193–198.

[39] H. Zhang, X. Gao, H. Luo *et al.*, "Sageroute: Synergistic analog routing considering geometric and electrical constraints with manual design compatibility," in *Proc. DATE*, 2023.

[40] Z. Shen, X. Tang, Z. Wu *et al.*, "A 9.7fj/conv.-step capacitive sensor readout circuit with incremental zoomed time domain quantization," in *Proc. CICC*, 2023.