

Special Research Session: PANDA: An LLM-Enhanced Performance-Driven Analog Design Framework Bridging Design Intent and Layout Generation

Haoyi Zhang¹, Weijian Fan¹, Xiaohan Gao¹, Bingyang Liu¹, Runsheng Wang^{1,2,3}, Yibo Lin^{1,2,3*}

¹School of Integrated Circuits, Peking University, Beijing, China

²Beijing Advanced Innovation Center for Integrated Circuits, Beijing, China

³Institute of Electronic Design Automation, Peking University, Wuxi, China

hy.zhang@pku.edu.cn, yibolin@pku.edu.cn

ABSTRACT

Traditional design of analog circuits heavily relies on manual interventions across topology, sizing, and layout, with prior automation addressing stages in isolation. In this work, we propose PANDA, an LLM-enhanced framework that bridges high-level design intent to final layout by actively managing cross-stage dependencies through guided topology synthesis, substructure-aware sizing, and constraint-driven layout generation. This shifts automation from algorithm-centric execution to intent-centric co-design, reducing turnaround time from days or weeks to hours while improving design performance. PANDA is released at <https://github.com/PKU-IDEA/PANDA>.

1 INTRODUCTION

Analog and mixed-signal (AMS) design automation is increasingly critical as circuit complexity grows, yet despite decades of analog CAD research, practical end-to-end automation remains limited. As illustrated in the traditional flow of Figure 1 (top), topology synthesis, sizing, and layout are handled in a loosely coupled manner, leaving three persistent gaps. First, conventional algorithms operate without understanding design intent, performing inefficient, directionless exploration that often misses optimal regions [7, 9–11, 15, 28, 31, 38, 42, 45]. Second, schematic and layout stages are isolated: sizing is optimized without layout awareness, while layout tools lack structured links to circuit-level decisions, breaking optimization loops [3, 21, 23, 24, 30, 33, 40, 47]. Third, existing flows cannot autonomously iterate based on post-layout simulation results; manual intervention is required to extract parasitics, re-evaluate performance, and adjust parameters, preventing true performance-driven refinement.

Prior research has made important advances in individual stages, including topology synthesis [25, 29, 48], transistor sizing [4, 5, 16, 32, 35], placement [2, 12, 26, 37, 49], and routing [6, 39, 43, 44]. Yet these efforts remain fragmented, and the integration of high-level design intent with physical implementation is

*Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
DAC'26, July 2026, Long Beach, CA, USA
© 2026 Association for Computing Machinery.

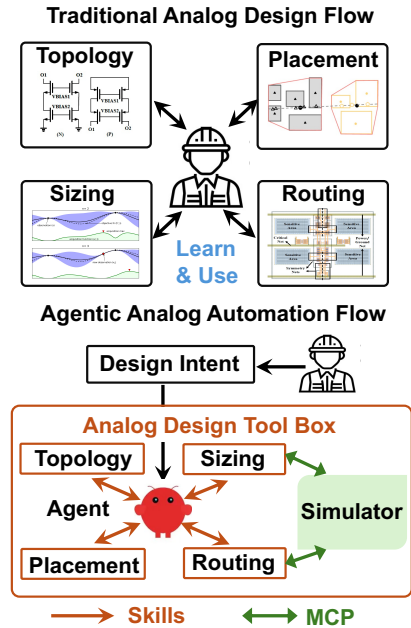


Figure 1: Comparison between traditional automated analog design flow and the agentic flow in PANDA.

still missing. Recent large language models (LLMs) offer new opportunities to bridge these gaps [8, 13, 17, 18, 20, 22, 27, 34, 36, 41], but intent understanding alone is insufficient without a framework that preserves cross-stage dependencies and grounds generated plans in deterministic tool executions.

PANDA addresses these challenges by employing an LLM as a design coordinator that interprets intent, manages cross-stage dependencies, and drives iterative improvement. As shown in the proposed agentic flow of Figure 1 (bottom), PANDA unifies topology synthesis, analog sizing, placement, and routing into a coherent pipeline. The LLM translates specifications into structured constraints so that sizing follows topology, placement respects device-level requirements, and routing receives consistent interface definitions. After layout, PANDA automatically extracts parasitics and feeds post-layout performance back to the LLM, enabling it to select the best implementation and iteratively refine the design. By bridging schematic and layout with a semantically aware coordinator, PANDA shifts analog automation from isolated, algorithm-centric execution to an intent-centric, self-improving flow.

In summary, this work makes four key contributions:

- **Agent Flow:** Agentic analog design flow that connects high-level design intent to final physical implementation

through four coordinated stages, preserving the dependency chain from circuit semantics to executable layout actions.

- **Topology Synthesis:** LLM-guided topology synthesis that translates compact design intent and specifications into explicit structural artifacts, providing a stable schematic representation for downstream optimization.
- **Analog Sizing:** Substructure-aware analog sizing using a multi-stage hierarchical Bayesian optimization engine that resolves device parameters from the generated topology and emits transistor-level artifacts directly consumable by layout.
- **Layout Generation:** Constraint-driven layout generation that combines placement and routing with schematic-layout synchronization, converting structural and sizing information into geometry-aware constraints and executable back-end actions.

Based on these contributions, PANDA demonstrates a unified path toward practical analog design automation, reducing turnaround time from days or weeks to hours while improving design performance. By bridging design intent and physical realization with an LLM-coordinated flow, PANDA represents a significant step toward closing the loop in analog design automation.

2 BACKGROUND AND MOTIVATION

Analog automation remains substantially less mature than digital design automation because the information required for design closure is distributed across multiple heterogeneous stages. While digital flows benefit from standardized abstractions and clean stage interfaces, analog design must continuously reconcile circuit intent, device-level performance, matching constraints, parasitic effects, and layout regularity.

2.1 Breakpoints in Analog Design Closure

Conventional analog flows contain three persistent breakpoints [7, 11, 19, 40]. First, the transition from user-level design intent to a feasible circuit topology is still highly manual and experience-driven. Second, transistor sizing is often optimized in isolation, making it difficult to transfer performance-critical information to later physical stages. Third, placement and routing require explicit geometric and electrical constraints, such as symmetry, common-centroid structures, and consistent pin definitions, which are rarely generated in a unified way from earlier design decisions. As a result, the flow is often broken into loosely coupled steps that require repeated manual repair.

2.2 From Agent Reasoning to Executable Flows

Recent agentic systems suggest that LLM-based workflows become substantially more reliable when reasoning is decomposed into reusable task units and connected to explicit tool interfaces [1, 17]. In this context, a *Skill* can be viewed as a stage-specialized capability that encapsulates prompting, context construction, and post-processing for a particular design task, while an MCP-style interface acts as a normalized protocol for invoking heterogeneous external tools and services. For analog automation, these abstractions are attractive because they separate semantic planning from physical execution without breaking the underlying stage dependencies.

PANDA adopts this perspective in a lightweight way. Skills provide structured reasoning entry points for topology, sizing, placement, and routing, whereas MCP-style invocations provide stable execution channels to the back-end engines. We introduce these concepts here as background because they motivate how PANDA connects LLM guidance with deterministic analog design closure; their concrete role in the framework is detailed in Section 3.

3 PANDA METHODOLOGY

PANDA is designed as an integrated framework that connects high-level design intent to final analog layout through four tightly coupled stages: topology synthesis, analog sizing, placement, and routing. Across these stages, the LLM serves as a planner and constraint generator, while the back-end engines execute physically grounded design actions. The methodology is therefore centered on progressive context refinement: each stage transforms design intent into more concrete structural, parametric, and geometric representations until executable layout actions are obtained.

3.1 Framework Overview

Figure 2 presents the overall PANDA workflow. The framework takes compact user inputs (`design_intent` and `design_spec`) and drives a four-stage refinement chain: topology synthesis, analog sizing, constraint-driven placement, and routing with schematic-layout synchronization. Rather than treating these stages as isolated tools, PANDA organizes them as a dependency-preserving pipeline in which each stage consumes structured outputs from the previous one and emits execution-ready artifacts for the next.

At the control level, PANDA separates semantic planning from deterministic execution. The LLM is responsible for interpreting intent, selecting stage-specific actions, and generating machine-readable constraints, while specialized back-end engines perform simulation, optimization, placement, and routing. This division ensures that high-level reasoning remains flexible, but physically grounded decisions are always produced by domain tools with explicit interfaces.

To make this interaction reproducible, PANDA implements the workflow as reusable *Skills* connected through explicit hand-off artifacts. As summarized in Table 1, these Skills cover generation, decision, planning, and execution tasks across all stages. Together, they form a closed-loop automation process that preserves cross-stage consistency and supports iterative improvement from post-layout feedback.

3.2 Topology Synthesis

In the first stage, PANDA invokes *AnalogXpert* to synthesize circuit topology as subcircuit-level SPICE generation from structured requirements (function, performance targets, and interface constraints) [46]. The agent narrows the search space with an extensible subcircuit library, constructs connectivity with iterative proofreading, and outputs a practical `.SUBCKT` representation. PANDA then parses this result into structured artifacts (netlist, device relations, and terminal mappings) and stores them in `topology.json`; if generation is invalid, it falls back to the closest valid subcircuit combination and re-runs refinement to recover a consistent topology for downstream stages.

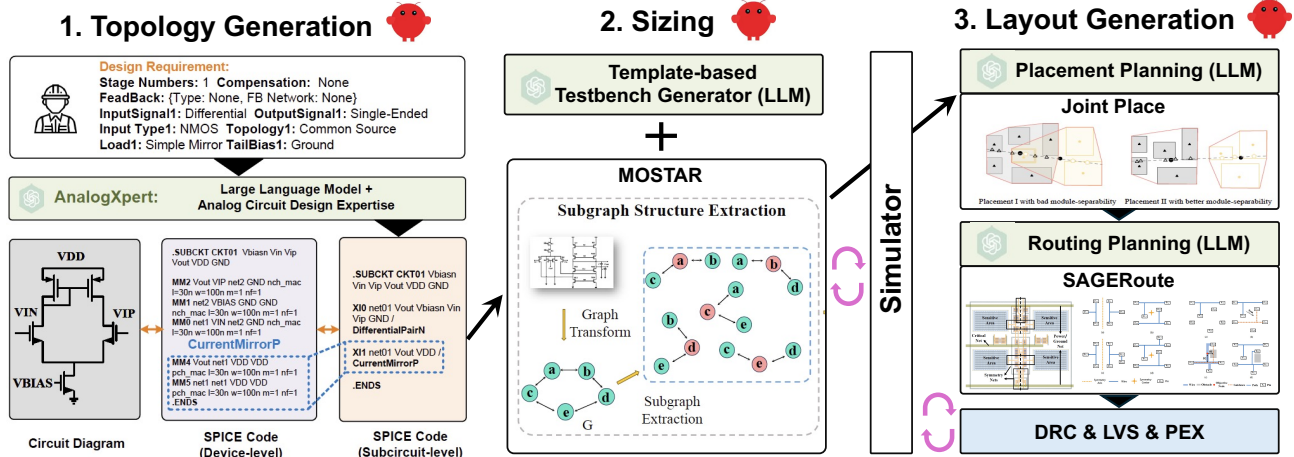


Figure 2: Overview of PANDA methodology.

Table 1: Abstracted task Skills in PANDA.

Task	Role	Main output
topology_generation	Generates the initial circuit topology from intent and specs.	topology.json
post_topology_decision	Dispatches downstream tasks and derives stage-specific hints.	planning JSONs
sizing	Resolves device parameters for target performance.	sizing.json
placement_planning	Builds placement constraints such as grouping and symmetry.	placement_pr.json
placement	Executes physical placement using planned constraints.	pl.json
routing_planning	Prepares routing directives and interface constraints.	routing_pr.json
routing	Executes detailed routing and final layout connectivity.	rt.json

3.3 Analog Sizing

With topology fixed, PANDA performs transistor-level sizing as constrained optimization over parameters such as width, length, finger number, and multipliers. The framework uses *MOSTAR*, a multi-stage hierarchical Bayesian optimizer that combines substructure-aware modeling (via L2G-GNN), symmetric constraints, additive Gaussian processes, and stage-adaptive acquisition to handle high-dimensional black-box objectives efficiently [14]. Optimization iteratively evaluates candidate designs and updates the surrogate model, and the final execution-ready parameter set is saved as *sizing.json* for subsequent physical layout stages.

3.4 Constraint-Driven Placement

In placement, PANDA translates schematic structure and layout intent into geometry using an AMS joint placement framework that co-optimizes area/wirelength proxies, symmetry and collinearity constraints, and signal/power-flow patterns while explicitly considering hierarchy (e.g., subtree and sibling relations) [15]. It then applies hierarchical legalization to remove overlaps and produce a compact routable solution, and stores the

final placement artifacts (including *pl.json*) for consistent downstream routing and quality evaluation.

3.5 Routing and Layout Synchronization

In the final stage, PANDA aggregates topology, sizing, and placement outputs to generate routing directives and executes detailed routing with *SAGERoute*/*SAGERoute2.0* under unified design-rule and interface constraints [42, 45]. To maintain schematic-layout consistency, it synchronizes the LVS pin list in *template_json/allconfig.json* from topology-derived interfaces, iteratively updates routing configurations, and runs *PEX* after each attempt to obtain parasitic-aware performance/yield feedback; these metrics are then fed back to the LLM to select and refine the best feasible routed solution [20].

4 EXPERIMENTAL RESULTS

We evaluate PANDA from two perspectives: execution transparency and design quality. In addition to reporting final layout and circuit performance, we show representative intermediate prompts and structured Skill outputs that reveal how the flow progresses from user intent to executable layout actions. We demonstrate PANDA on two cases, *OTA* and *COMP*. Due to page limits, we present the *OTA* flow in detail and summarize *COMP* with key metrics.

For *OTA*, we use the design request “Design a three-stage *OTA* with differential input and single output, targeting high gain and robust phase margin for sensor readout,” with *design_spec* *circuit_type*=ota, *target_gain_db*=75, and *target_ugbw_mhz*=8. For *COMP*, we evaluate a StrongARM comparator case and summarize the post-layout power and delay results reported in Table 2.

4.1 Execution Flow

PANDA executes the design task as a staged pipeline from *design_intent* and *design_spec* to final layout. The flow first generates a topology artifact, then derives sizing parameters, placement constraints, and routing directives, with each stage emitting an explicit machine-readable output for the next one. These intermediates are stored in attempt-specific files such as *topology.json*, *sizing.json*, *pl.json*, and *rt.json*, which makes the run traceable and easy to inspect.

To preserve cross-stage consistency, PANDA synchronizes downstream inputs directly from upstream artifacts instead of reconstructing them from free-form text. For example, routing configurations inherit interface information such as `lvs_pins` from topology outputs or parsed `.SUBCKT` definitions, while placement-derived symmetry information is persisted as structured constraints for later stages. This organization keeps the execution chain reproducible and maintains alignment between schematic intent and physical implementation.

In our experiments, the end-to-end runtime for a full PANDA run is at the *several-hours* level, whereas comparable manual analog layout iterations typically require *several days*. This gap highlights the practical turnaround advantage of the staged and observable PANDA flow.

4.2 Prompt and Skill Examples

Beyond final artifacts, PANDA also records representative prompt-level intermediates that show how design intent is progressively translated into structured downstream actions. Rather than presenting full prompt transcripts, we highlight three concise examples that correspond to different abstraction levels in the flow.

Example Prompt and Skill Interactions in PANDA

Design Intent:

The flow begins from a compact user-level request: design a three-stage OTA with differential input and single output, targeting high gain and robust phase margin for sensor readout. The corresponding structured context in `flow_request.json` includes `target_gain_db=75` and `target_ugbw_mhz=8`. PANDA interprets this request as a flow-level objective and initializes the task scope, context-loading policy, and retry budget.

Post Topology Decision:

After topology generation, PANDA invokes the post-topology decision Skill to convert the topology summary into downstream actions. For the generated three-stage OTA, the Skill schedules the next agents for sizing, placement, and routing; prepares files such as `sizing_spec.json`, `placement_constraints.json`, and `routing_rules.json`; and derives guidance including symmetry pair [X11, X12], critical devices [X17, X18], and sensitive nets [OUT1, OUT2, OUT3].

Placement Planning:

At the placement stage, the placement-planning Skill focuses on analog-layout-specific constraints including grouping, symmetry, matching, and whitespace reservation. The planning prompt explicitly requires that all PMOS devices be grouped, while NMOS devices may remain ungrouped. The resulting `placement_pr.json` records clustered placement objects, symmetric pairs such as [MM25, MM26] and [MM39, MM40], whitespace settings, and an automatic consistency check that reports full PMOS coverage.

These examples show that PANDA does not rely on opaque prompt chaining. Instead, each prompt is tied to a stage-specific Skill and materialized as a concise structured object that can be executed, checked, and propagated to later stages.

4.3 Pre-Layout and Post-Layout Performance

To evaluate design quality, Table 2 reports pre- and post-layout results for both OTA and COMP. For OTA, the post-layout results (via PEX) retain high gain (76.11 dB) and robust phase margin (65.3°), which are consistent with the design intent of high-gain and stable sensor-readout operation. The UGB decreases from 10.84 MHz to 4.192 MHz after layout parasitics are included, so the frequency target is not fully maintained post-layout even though the main amplification behavior is preserved. For COMP, no explicit quantitative performance targets are specified in the design request, so we report power and delay as representative implementation metrics. The post-layout result shows 606.6 nW

power and 1.447 ns delay, compared with 469.9 nW and 1.0 ns pre-layout, indicating the expected parasitic-induced overhead while remaining within a practical operating range for a functional StrongARM comparator implementation. Overall, the post-layout results show that PANDA can carry both designs from intent to physically realized implementations with clear and traceable performance tradeoffs.

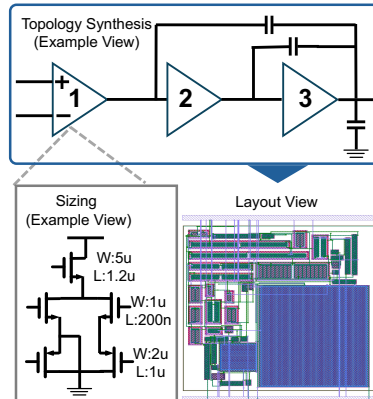


Figure 3: Generated layout result for the OTA case.

Table 2: Pre-layout and post-layout performance comparison for the OTA and COMP case studies.

Design	Metric	Design intent	Pre-layout	Post-layout
OTA	Gain (dB)	> 75	79.07	76.11
	PM (°)	> 60	67.47	65.3
	UGB (MHz)	> 8	10.84	4.192
	Power (μ W)	–	411.2	435.2
COMP	Power (nW)	–	469.9	606.6
	Delay (ns)	–	1.0	1.447

4.4 Attempt-Level Analysis

PANDA is designed to be both observable and executable: it logs stage transitions, invocation status, and generated artifacts, and stores each trial as a consistent snapshot (topology, sizing, placement, routing, and configuration). This trial-level organization enables direct comparison across candidate solutions. For each completed attempt, PANDA runs *PEX* and post-layout simulation, then uses the resulting electrical metrics together with physical check outputs to select the best end-to-end implementation for the target specifications.

5 CONCLUSION AND FUTURE ROADMAP

This paper presented PANDA, an LLM-enhanced performance-driven analog design framework that bridges design intent and layout generation. By integrating topology synthesis, analog sizing, placement, and routing into a coherent flow, PANDA turns high-level requirements into executable circuit and layout actions while preserving cross-stage consistency.

Future work will focus on strengthening the learning capability and portability of the framework, including more powerful planning models, tighter coupling between performance estimation and physical optimization, and cleaner service interfaces for back-end tools. These improvements will further advance PANDA toward a more scalable and general analog design automation framework.

REFERENCES

- [1] Jinheon Baek et al. 2024. ResearchAgent: Iterative Research Idea Generation over Scientific Literature with Large Language Models.
- [2] F. Balasa et al. 2002. Efficient solution space exploration based on segment trees in analog placement with symmetry constraints. In *IEEE/ACM International Conference on Computer Aided Design, 2002. ICCAD 2002*. 497–502.
- [3] S. Bhattacharya et al. 2005. Template-driven parasitic-aware optimization of analog integrated circuit layouts. *Proceedings. 42nd Design Automation Conference, 2005.* (2005), 644–647.
- [4] Weidong Cao et al. 2022. Domain knowledge-infused deep learning for automated analog/radio-frequency circuit parameter optimization. In *DAC (DAC '22)*. Association for Computing Machinery, New York, NY, USA, 1015–1020.
- [5] Chen Chen et al. 2022. High-Dimensional Bayesian Optimization for Analog Integrated Circuit Sizing Based on Dropout and gm/ID Methodology. *TCAD* 41, 11 (2022), 4808–4820.
- [6] Hao Chen et al. 2020. Toward silicon-proven detailed routing for analog and mixed-signal circuits. In *ICCAD*. 1–8.
- [7] Hao Chen et al. 2021. MAGICAL: An Open-Source Fully Automated Analog IC Layout System from Netlist to GDSII. *IEEE Design & Test* 38, 2 (2021), 19–26.
- [8] Zihao Chen et al. 2024. Artisan: Automated Operational Amplifier Design via Domain-specific Large Language Model. In *DAC (DAC '24)*. Association for Computing Machinery, New York, NY, USA, 6.
- [9] J.M. Cohn et al. 1991. KOAN/ANAGRAM II: new tools for device-level analog placement and routing. *IEEE Journal of Solid-State Circuits* 26, 3 (1991), 330–342.
- [10] J. Crossley et al. 2013. BAG: A designer-oriented integrated framework for the development of AMS circuit generators. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2013-11). 74–81. ISSN: 1558-2434.
- [11] Tommoy Dhar et al. 2020. The ALIGN open-source analog layout generator: v1.0 and beyond. In *Proceedings of the 39th International Conference on Computer-Aided Design (2020-11-02)*. ACM, 1–2.
- [12] Tommoy Dhar et al. 2022. A Charge Flow Formulation for Guiding Analog/Mixed-Signal Placement. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 148–153.
- [13] Abhimanyu Dubey et al. 2024. The Llama 3 Herd of Models. *arXiv* (2024).
- [14] Weijian Fan et al. 2026. MOSTAR: Multi-Stage Hierarchical Bayesian Optimization for Substructure-Aware High-Dimensional Analog Circuit Sizing. In *2026 31st Asia and South Pacific Design Automation Conference (ASP-DAC)*. 653–659.
- [15] Xiaohan Gao et al. 2024. Joint Placement Optimization for Hierarchical Analog/Mixed-Signal Circuits. In *2024 ACM/IEEE International Conference on Computer Aided Design (ICCAD)*. 1–9.
- [16] Xiaohan Gao et al. 2024. Post-layout simulation driven analog circuit sizing. *SCIS* 67, 4 (2024), 142401.
- [17] Zhuolun He et al. 2024. ChatEDA: A Large Language Model Powered Autonomous Agent for EDA.
- [18] Yao Lai et al. 2024. AnalogCoder: Analog Circuit Design via Training-Free Code Generation. *arXiv* (2024).
- [19] Mark Po-Hung Lin et al. 2017. Parasitic-Aware Common-Centroid Binary-Weighted Capacitor Layout Generation Integrating Placement, Routing, and Unit Capacitor Sizing. *IEEE*, 36, 8 (2017), 1274–1286.
- [20] Bingyang Liu et al. 2025. LayoutCopilot: An LLM-Powered Multi-Agent Collaborative Framework for Interactive Analog Layout Design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2025), 1–1.
- [21] Mingjie Liu et al. 2021. Parasitic-Aware Analog Circuit Sizing with Graph Neural Networks and Bayesian Optimization. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2021-02-01). IEEE, 1372–1377.
- [22] Mingjie Liu et al. 2024. ChipNeMo: Domain-Adapted LLMs for Chip Design.
- [23] N. Lourenco et al. 2006. LAYGEN - Automatic Layout Generation of Analog ICs from Hierarchical Template Descriptions. In *2006 Ph.D. Research in Microelectronics and Electronics* (2006). IEEE, 213–216.
- [24] Nuno Lourenço et al. 2015. Layout-Aware Sizing of Analog ICs using Floorplan & Routing Estimates for Parasitic Extraction. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015* (2015). IEEE Conference Publications, 1156–1161.
- [25] Jialin Lu et al. 2022. Topology Optimization of Operational Amplifier in Continuous Space via Graph Embedding. In *DATE (2022-03-14)*. IEEE, 142–147.
- [26] Qiang Ma et al. 2011. Simultaneous Handling of Symmetry, Common Centroid, and General Placement Constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30, 1 (2011), 85–95.
- [27] Aman Madaan et al. 2023. Self-Refine: Iterative Refinement with Self-Feedback. *arXiv* (2023).
- [28] Ricardo M. F. Martins et al. 2023. Analog Integrated Circuit Routing Techniques: An Extensive Review. *IEEE Access* 11 (2023), 35965–35983.
- [29] Markus Meissner et al. 2015. FEATS: Framework for Explorative Analog Topology Synthesis. *IEEE*, 34, 2 (2015), 213–226.
- [30] Hung-Chih Ou et al. 2016. Layout-Dependent Effects-Aware Analytical Analog Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35, 8 (2016), 1243–1254.
- [31] J. Rijmenants et al. 1989. ILAC: an automated layout tool for analog CMOS circuits. *IEEE Journal of Solid-State Circuits* 24, 2 (1989), 417–425.
- [32] Keertana Settalur et al. 2022. Automated Design of Analog Circuits Using Reinforcement Learning. *IEEE*, 41, 9 (2022), 2794–2807.
- [33] Mohammad Torabi et al. 2018. Electromigration- and Parasitic-Aware ILP-Based Analog Router. *IEEE*, 26, 10 (2018), 1854–1867.
- [34] Hugo Touvron et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv* (2023).
- [35] Hanrui Wang et al. 2020. GCN-RL Circuit Designer: Transferable Transistor Sizing with Graph Neural Networks and Reinforcement Learning. In *DAC*. 1–6.
- [36] Jason Wei et al. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv* (2023).
- [37] Biying Xu et al. 2019. Device Layer-Aware Analytical Placement for Analog Circuits. In *Proceedings of the 2019 International Symposium on Physical Design (ISPD '19)*. Association for Computing Machinery, New York, NY, USA, 19–26.
- [38] Biying Xu et al. 2019. MAGICAL: Toward Fully Automated Analog IC Layout Leveraging Human and Machine Intelligence: Invited Paper. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–8.
- [39] Peng Xu et al. [n. d.]. Performance-driven Analog Routing via Heterogeneous 3DGNN and Potential Relaxation. ([n. d.]).
- [40] Peng Xu et al. 2024. Performance-Driven Analog Layout Automation: Current Status and Future Directions (Invited Paper). In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)* (2024-01-22). IEEE, 679–685.
- [41] Yuxuan Yin et al. 2024. ADO-LLM: Analog Design Bayesian Optimization with In-Context Learning of Large Language Models.
- [42] Haoyi Zhang et al. 2023. SAGERoute: Synergistic Analog Routing Considering Geometric and Electrical Constraints with Manual Design Compatibility.
- [43] Haoyi Zhang et al. 2023. SAGERoute: Synergistic Analog Routing Considering Geometric and Electrical Constraints with Manual Design Compatibility. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1–6.
- [44] Haoyi Zhang et al. 2024. SAGERoute 2.0: Hierarchical Analog and Mixed Signal Routing Considering Versatile Routing Scenarios. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1–6.
- [45] Haoyi Zhang et al. 2024. SAGERoute2.0: Hierarchical Analog and Mixed Signal Routing Considering Versatile Routing Scenarios. In *2024 Design, Automation, Test in Europe Conference Exhibition (DATE)*. 1–6.
- [46] Haoyi Zhang et al. 2025. AnalogXpert: Automating Analog Topology Synthesis by Incorporating Circuit Design Expertise into Large Language Models. In *2025 International Symposium of Electronics Design Automation (ISED)*. 772–777.
- [47] Lihong Zhang et al. 2008. Parasitic-Aware Optimization and Retargeting of Analog Layouts: A Symbolic-Template Approach. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 5 (2008), 791–802.
- [48] Zhenxin Zhao et al. 2020. An Automated Topology Synthesis Framework for Analog Integrated Circuits. *IEEE*, 39, 12 (2020), 13.
- [49] Keren Zhu et al. 2020. Effective Analog/Mixed-Signal Circuit Placement Considering System Signal Flow. In *2020 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. 1–9.