LayoutCopilot: LLM-Empowered Analog Layout Design towards Enhanced Human-Machine Interaction

Bingyang Liu[1], Haoyi Zhang[2], Xiaohan Gao[3], Xiyuan Tang[4,2], Yibo Lin[2,5,6*], Runsheng Wang[2,5,6], Ru Huang[2,5,6]
[1]School of EECS, [2]School of Integrated Circuits, [3]School of Computer Science, [4]Institute for Artificial Intelligence, Peking University
[5]Beijing Advanced Innovation Center for Integrated Circuits [6]Institute of EDA, Peking University, Wuxi, China
{lby-1132, hy.zhang}@stu.pku.edu.cn, {xiaohangao, xitang, yibolin, r.wang, ruhuang}@pku.edu.cn

***Abstract*—Analog and mixed-signal circuits are crucial for interfacing digital systems with the real world, yet the layout design remains manual and highly labor-intensive. Fully automated tools for layout design have made significant progress in easing this burden, but they often restrict flexibility and designer control. Interactive design flows combine the strengths of both manual and automated design; however, designers still face challenges in human-machine interaction, such as complex command sets and manual code writing. In this paper, we introduce LayoutCopilot, an LLM-empowered interactive layout design framework that addresses this challenge by enabling the translation of high-level design intents expressed in natural language into actionable commands. It also incorporates automated constraint extraction, reducing repetitive tasks and enhancing interaction between designers and the tool. Our experiments demonstrate that this framework undergoes validation for syntactic and functional correctness and is successfully applied to real-world analog design tasks, from constraint extraction to layout refinement, achieving efficient designers' involvement with reduced manual efforts.**

***Index Terms*—analog layout synthesis, LLM-empowered, human-machine interaction**

I. Introduction & Background

Analog and mixed-signal circuits play a critical role in interfacing digital systems with the real world, yet the layout design remains largely manual. This reliance on human expertise not only extends design cycles but also increases costs. Moreover, performance degradation occurs during the transition from schematic to layout due to parasitics and other complexities, requiring iterative adjustments to optimize performance. These challenges make it difficult to avoid the final fine-tuning of the layout designers, as illustrated in Figure 1 A.

Efforts to automate analog layout design can be categorized into three approaches. Industrial tools such as Cadence Virtuoso [1] offer scripting capabilities to automate repetitive tasks, but steep learning curves hinder widespread adoption. In the academic realm, a series of analog placement and routing algorithms have been proposed to automate layout design [2]–[14], leading to the development of mature fully-automated tools such as MAGICAL [15]–[17] and ALIGN [18]. However, these tools often lack design flexibility, limiting opportunities for designer intervention, as illustrated in Figure 1 B. In response to these limitations, interactive design tools [19]–[21] have emerged, enabling designers to adjust layouts in real time through command-based interfaces. These tools try to bridge the gap between manual and automated processes. However, the complexity of command sets can increase cognitive load, and coding the commands remains a repetitive task.

Meanwhile, the rise of Large Language Models (LLMs) is offering new possibilities for analog Electronic Design Automation (EDA) tools. In analog EDA, LLMs are being applied to tasks such as sizing [22]–[26], topology synthesis [27], [28], and layout generation [26], [29]. Moreover, conversational interfaces such as [30], [31] have simplified interactions with EDA tools, allowing designers to engage with complex workflows through natural language, reducing the need for extensive command scripting. Recent work like [32] also demonstrates how LLMs can facilitate the efficient design of complex analog circuits by extracting and porting design knowledge from literature, enhancing process and performance.

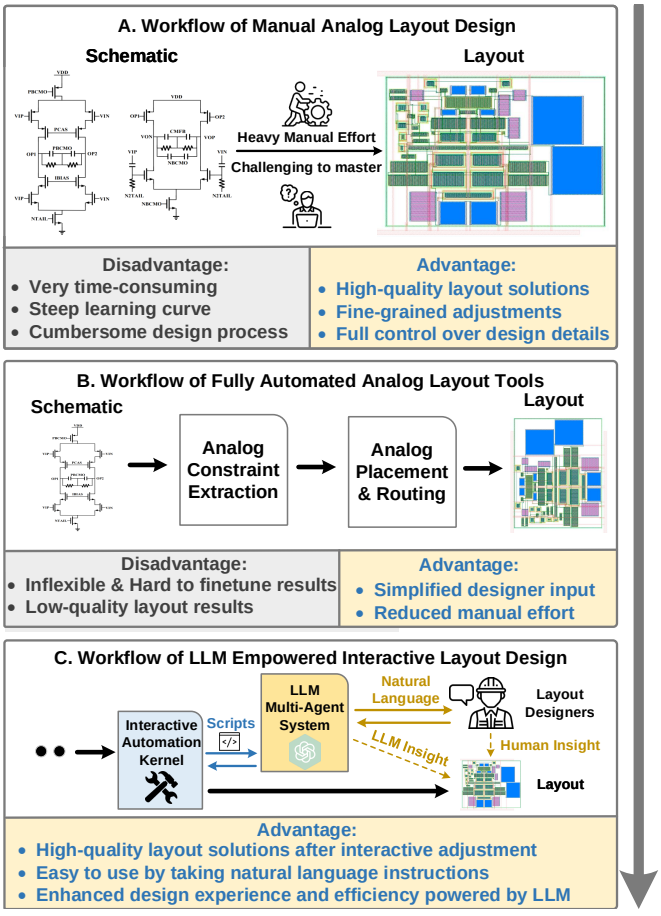


Fig. 1: Workflows of manual, fully automated, and LLM-powered interactive analog layout design tools (LayoutCopilot), comparing key advantages and disadvantages.

In this paper, we introduce LayoutCopilot, an LLM-empowered framework for interactive analog layout design, aimed at enhancing human-machine interaction [33]. As illustrated in Figure 1 C, our approach reduces the manual effort required in interactive design by translating natural language design intents into layout modification commands. LayoutCopilot also incorporates automated constraint extraction to reduce repetitive tasks further and enhance interaction. Experimental results demonstrate that the design intent translation was validated for correctness and functionality, accurately converting

*Corresponding author: Yibo Lin (yibolin@pku.edu.cn)

intent into constraints across various cases. In real-world analog design tasks, LayoutCopilot proved highly effective from constraint extraction to layout refinement. It increases designer involvement while reducing manual effort, accelerating the design process, and improving efficiency.

The rest of the paper is organized as follows, II explains the details of the proposed framework; III validates the framework with experimental results; IV concludes the paper.

II. Framework

The workflow of LayoutCopilot consists of three main phases: constraint extraction through LLM, initial solution generation via a placement and routing (P&R) kernel, iterative optimization using the interactive kernel, and the Abstract/Concrete Request Processors, as illustrated in Figure 2. The inputs to the entire flow—SPICE netlist, instructions, knowledge, and design intents—are passed to LayoutCopilot during both the initial solution generation and iterative optimization phases.

A. Constraint Extraction

In the first phase, LayoutCopilot preprocesses the netlist to extract design constraints that guide the subsequent P&R stages. The key preprocessing step is constructing an intermediate representation, called a signal graph, which reorganizes the connection relationship of devices to make it more interpretable for the LLM. As shown in Figure 3, the signal graph is built by focusing on the source (S) and drain (D) connections of transistors, while ignoring gate and bulk connections. This allows for the creation of a graph that captures the paths from VDD to GND, grouping components accordingly. Each transistor in the netlist as well as the ground and power nets are represented as nodes. The purpose of this graph is to provide a structured and readable format for the LLM, which improves its ability to extract relevant constraints. After the building of the signal graph, devices in the netlist are grouped based on their paths from VDD to GND. This grouping allows the LLM to better understand the structure of the circuit and extract constraints that are crucial for layout quality.

For instance, as shown in Figure 3, in the case of an OTA, after preprocessing the netlist and determining the number of stages based on the IO nets, symmetry analysis is performed within each stage. This analysis involves identifying symmetrical pairs of devices, such as differential pairs, current mirrors, and other elements sized the same. Using the knowledge of analog circuit design, layout design principles, and provided examples, the LLM recognizes symmetrical structures and labels them accordingly. While this paper focuses on symmetry extraction, similar methods could be applied to other tasks. With the right preprocessing (such as the signal graph) and appropriate instructions, knowledge, and examples, LLMs could be expanded to extract different types of constraints, enabling broader applications in layout design.

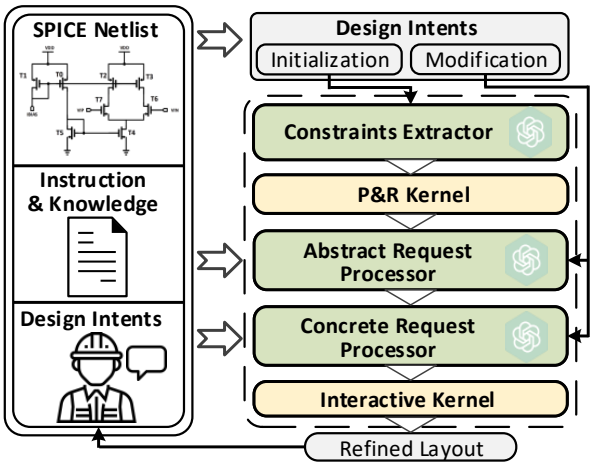


Fig. 2: Interactive layout design flow with LayoutCopilot, which includes constraint extraction, placement and routing (P&R) kernel, and an interactive kernel for iterative layout modifications.

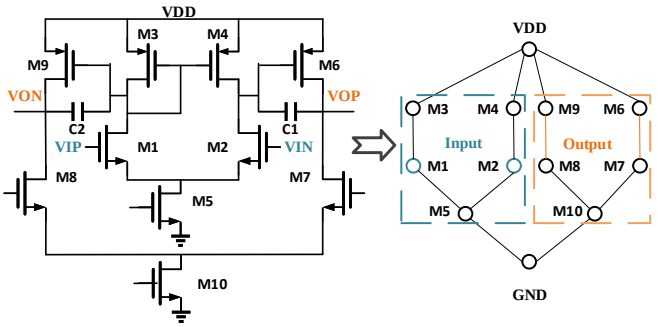


Fig. 3: The process of generating a signal graph from the netlist, as illustrated for an OTA circuit. Blue represents input-related devices and nets, while orange represents output-related devices and nets.

B. P&R and Interactive Kernels

The P&R kernel [34]–[37] is responsible for generating an initial layout by utilizing both the netlist and specified constraints. It generates the initial layout by solving a nonlinear optimization problem, where the objective function typically minimizes the half-perimeter wire length (HPWL) while integrating several basic constraints [34]. The routing process employs an A* algorithm-based router, which finds the shortest paths while considering additional routing constraints such as symmetry, routing congestion, and electromigration [35]–[37].

The interactive kernel [19], [20] facilitates real-time, user-driven layout adjustments, building on earlier P&R kernels. This kernel allows designers to interact with the layout through commands and a graphical user interface (GUI). Designers can issue placement commands such as `move`, `swap`, and `symAdd` to adjust component positions and apply symmetry or array constraints. Similarly, routing can be modified using commands like `reroute`, `wireWidth`, and `netPriority` enabling adjustments to wire paths, widths, and spacing for performance tuning [19], [20]. The interactive kernel bridges the gap between fully automated tools and manual adjustments, offering designers greater control and flexibility while reducing repetitive tasks.

C. Abstract/Concrete Request Processor

After generating the initial layout, further interactive optimization is necessary as the initial solution may not fully satisfy all design-specific performance requirements. The Abstract/Concrete Request Processor manages this iterative refinement by translating high-level design intents into executable layout commands. It leverages a multi-agent system to handle the complexity of converting abstract requests, such as improving symmetry or matching, into concrete commands for the layout tool.

The Abstract Request Processor is structured as a multi-agent system that handles high-level, conceptual requests from designers. These requests often require deeper analysis and knowledge retrieval before they can be transformed into actionable commands. The

system includes a task classifier agent that determines whether a designer's input is an abstract or concrete request. For abstract tasks, it passes the input to the Analyzer Agent, which draws on a dedicated knowledge base containing analog design principles and previous solutions. The Analyzer Agent breaks down abstract design intents, such as "enhance matching" into specific layout strategies, including modifying placement and routing. After the initial solution is proposed, the Solution Refiner Agent incorporates designer feedback to refine and optimize the proposed modifications, ensuring that the final solution is closely aligned with the designer's intent. Finally, the Solution Adapter Agent converts the determined modifications into concrete requests using the netlist information. This allows the abstract intent to be transformed into a series of executable steps, ready for the next stage.

The Concrete Request Processor focuses on translating the concrete tasks into executable layout commands. The Task Splitter Agent first decomposes each concrete request into a series of smaller, well-defined sub-tasks that correspond to specific commands in the layout tool. The Code Generator Agent plays a central role here, ensuring that each command adheres to the syntax and operational logic required by the layout tools. Finally, the commands are combined to form code that directly influences placement and routing. The interactive kernel then takes in these commands to adjust the layout in real-time, and the system continues to iterate based on designer feedback to achieve the desired results.

D. Prompt Design and Configurations of Agents

Both the Abstract and Concrete Request Processors use prompt engineering techniques, such as chain-of-thought, control flow, and self-refine, to handle tasks efficiently. These shared strategies help agents break down complex tasks into manageable steps, respond correctly in complex scenarios, and validate outputs at each stage. On the other hand, the prompts are also tailored to the unique roles of each processor. In the Abstract Request Processor, agents act like experienced analog designers, focusing on high-level tasks and generating design suggestions; the Concrete Request Processor focuses on translating these refined suggestions into precise, executable commands. The prompts in this case emphasize correctness in syntax and logic, ensuring seamless integration into the layout flow. To illustrate the design of the prompt more concretely, we provide the Solution Refiner Agent's prompt (the most complex agent) as an example [1].

III. Experimental Results

In this section, we conduct experiments focusing on both concrete requests and a full design flow, demonstrating the accuracy and stability of LayoutCopilot in generating syntactically and logically correct commands, as well as its ability to handle comprehensive circuit design tasks. LayoutCopilot is compatible with various LLMs and layout tools, allowing for flexible deployment in different environments. For our experiments, we test LayoutCopilot with multiple versions of leading LLMs, including GPT-3.5 [38], GPT-4 [39], and Claude3 [40], to demonstrate its versatility.

A. Batch Testing For Concrete Request Processor

To evaluate the robustness and accuracy of the Concrete Request Processor, we conducted a series of tests focusing on two key aspects: Sanity Check and Functionality Check.

[1]https://github.com/PKU-IDEA/LayoutCopilot-Prompt-Example

TABLE I: Sanity checks and comparison for single-agent with instruction vs. multi-agent.

Category	**Single-agent**			
	GPT-3.5	GPT-4	Claude 3	**Avg.**
Formatting	71.14%	90.91%	99.25%	87.20%
Validity	91.36%	93.60%	95.44%	93.47%
Syntax	67.11%	88.87%	95.24%	83.74%
Logic	66.44%	83.04%	91.67%	80.38%
Overall	66.27%	82.91%	90.77%	79.98%
Category	**Multi-agent**			
	GPT-3.5	GPT-4	Claude 3	**Avg.**
Formatting	95.38%	99.76%	99.92%	**98.26%**
Validity	98.24%	99.28%	98.88%	**98.77%**
Syntax	92.65%	97.20%	96.96%	**95.60%**
Logic	91.24%	94.24%	98.80%	**94.76%**
Overall	90.92%	93.92%	**96.80%**	**93.75%**

1) Sanity Check: In the Sanity Check, we evaluated the ability of the processor to generate commands that adhere to syntax and logic rules. Out of the 1,250 requests, 1,134 were valid and could be transformed into commands requiring between 5 and 40 steps, while others were invalid and should be detected and flagged by LayoutCopilot. We tested both a single-agent and multi-agent configuration, where the multi-agent setup specialized each agent for different tasks as described in II-C, compared to the single-agent approach that merged all tasks into one LLM agent. Metrics and rules are the same as those defined in [33], which are omitted here due to space limitations. As summarized in Table I, the multi-agent configuration consistently outperformed the single-agent setup, demonstrating an overall correctness rate of 96.80% in handling concrete tasks when using Claude-3 [40], particularly in identifying and rejecting invalid requests. The results show that the multi-agent system is more effective in managing the complexity of concrete requests by ensuring commands are both syntactically logically correct [33].

2) Functionality Check: To further assess the functional accuracy of the generated commands, we conducted a Functionality Check on a random subset of 25 valid requests that had passed the sanity check. These cases were manually inspected to verify whether the commands generated by the Concrete Request Processor successfully executed the intended design tasks. The results were classified into three levels: Level A is functionally correct with clear reasoning. Level B has minor functional flaws but clear reasoning, allowing designers to fix issues through LLM interaction. Level C has both functional and

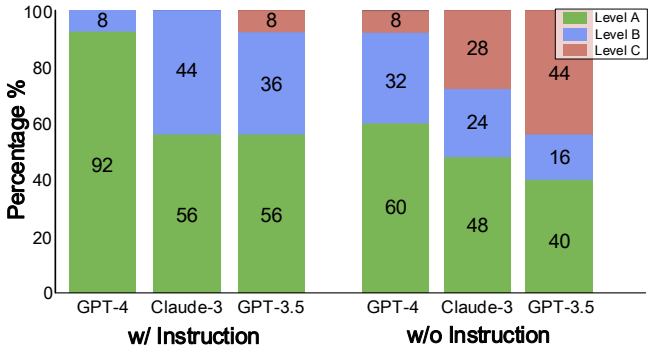
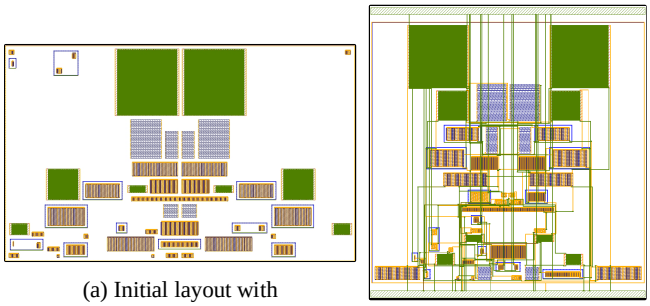


Fig. 4: Functionality check with different LLM engines across instructional conditions. The explanation of levels A, B, and C is provided in Section III-A2.



(a) Initial layout with constraints from LLMs

(c) After direct routing.

(b) After adding and removing some symmetry constraints.

(d) Adjusted the width of power and ground wires.

Fig. 5: Iterative layout optimization process for the OTA from the initial layout with constraints from LLMs to the final layout after routing.

conceptual flaws, offering little help in problem diagnosis. As shown in Figure 4, the instruction significantly reduced the proportion of Level C outputs, with most LLM engines producing Level A outputs in over 90% of cases when instructions are provided. Additionally, the percentage of C level under instructed conditions is near zero regardless of the LLM engine used. This demonstrates that the concrete task processor can get good results under conditions where various LLMs are used as engines, even if there are gaps in the capabilities of these LLM engines [33].

B. Case Study: Full Flow Demonstration

In this section, we demonstrate the full layout design process for an operational transconductance amplifier (OTA) using LayoutCopilot, focusing on constraint extraction, iterative layout refinement, and the post-simulation results. Another case for a comparator can be found in [33], which is omitted here due to space limitations.

For the OTA's gain path, we applied the constraint extractor, which combines a signal-graph approach with LLM-powered analysis. The layout generated after running the P&R kernel with automatically extracted constraints, shown in Figure 5 (a), demonstrates that most devices exhibit good symmetry. However, a few devices placed in the upper-left and upper-right corners of the layout represent an unreasonable placement pattern. This indicates that while the constraints extracted by the LLM are generally effective, further iterative optimization is needed.

The interactive layout optimization proceeded through multiple iterations, starting with the initial solution generated based on the LLM-extracted constraints. The process is visualized in four stages, as shown in Figure 5. We conveyed the aforementioned observations from Figure 5 (a) using natural language to the Abstract Request Processor and implemented the modification intent of adding and removing some of the symmetry constraints using the Concrete Request Processor. Figure 5 (b) shows the result after the modification, following which the P&R kernel was invoked to perform routing, as seen in Figure 5 (c). Finally, by adjusting the routing priorities, specifically increasing the width of the power and ground lines to improve performance, we obtained the final layout, shown in Figure 5 (d). The detailed dialog process can be found in [33]; due to space constraints, it is not elaborated here.

TABLE II: Comparison of performance metrics for schematic, MAGICAL [15]–[17] without constraints, initial layout with LLM constraints (placement shown in Figure 5 (a)), and the layout after interactive modification.

Metrics	Schematic	MAGICAL [15]–[17] w/o Constraints	Initial layout w/ LLM Constraints	Final
Gain (dB)	38.63	-8.75	**38.55**	38.26
UGB (MHz)	6.85	–	**4.65**	4.42
CMRR (dB)	–	27.3	42.9	**58.7**
PM (degree)	70.98	–	69.15	**76.28**

To evaluate the circuit's post-layout performance, we conducted simulations under TSMC 40nm technology using Cadence Virtuoso and Mentor Graphics Calibre. The performance is shown in Table II. In the layout generated by MAGICAL without constraints, the performance metrics show significant degradation due to excessive parasitics compared to the schematic. Due to excessive parasitic, the Gain was negative, the unity-gain bandwidth (UGB) and phase margin (PM) were both substantially poor, and the common-mode rejection ratio (CMRR) was significantly low. However, with the LLM-extracted constraints, the initial layout showed a marked improvement: the Gain was close to the schematic and after modification results, while the CMRR and PM were still below expected levels. Moreover, after applying interactive modifications, the optimized layout exhibited a minimal reduction in Gain and UGB while achieving substantial improvements in CMRR and PM. These results validate the effectiveness of our proposed flow in optimizing layouts through natural language interaction. The post-simulation improvements confirm that LayoutCopilot not only reduces the learning curve and coding effort for designers but also enables them to refine the layout more effectively, demonstrating its potential as a helpful tool for advanced interactive analog layout design.

IV. Conclusion

The proposed LLM-empowered Interactive Layout Design Framework, LayoutCopilot, effectively enhances human-machine interaction in analog layout design. By translating high-level design intents expressed in natural language into actionable layout modifications, LayoutCopilot addresses the human-machine interaction challenges in interactive design flows. The integration of automated constraint extraction reduces repetitive tasks, allowing designers to focus on more strategic aspects of the process. Our experiments validate the correctness of design intent translation and demonstrate LayoutCopilot's application in real-world analog tasks, from constraint extraction to layout refinement. This framework not only ensures a high level of designer involvement but also significantly reduces manual effort, enhancing both efficiency and design flexibility. Future work will focus on optimizing the translation process and expanding the framework's applicability to broader design workflows.

Acknowledge

This work is supported in part by the National Science Foundation of China (Grant No. 62141404, 62125401, 62034007), the Natural Science Foundation of Beijing, China (Grant No. Z230002), Grant No. QYJS-2023-2303-B, and the 111 project (B18001).

REFERENCES

[1] Cadence Design Systems, *Virtuoso Layout Suite*, Cadence Design Systems, Inc., San Jose, CA, USA, 2023.

[2] K. Lampaert *et al.*, "A performance-driven placement tool for analog integrated circuits," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 773–780, 1995.

[3] K. Lampaert *et al.*, "Analog routing for manufacturability," in *Proceedings of Custom Integrated Circuits Conference*, 1996, pp. 175–178.

[4] E. Malavasi *et al.*, "A routing methodology for analog integrated circuits." in *ICCAD*. Citeseer, 1990, pp. 202–205.

[5] L. Xiao *et al.*, "Practical placement and routing techniques for analog circuit designs," in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2010, pp. 675–679.

[6] B. Xu *et al.*, "Hierarchical and analytical placement techniques for high-performance analog circuits," in *Proceedings of the 2017 ACM on International Symposium on Physical Design*, 2017, pp. 55–62.

[7] B. Basaran *et al.*, "Latchup-aware placement and parasitic-bounded routing of custom analog cells," in *Proceedings of 1993 International Conference on Computer Aided Design (ICCAD)*. IEEE, 1993, pp. 415–421.

[8] H.-C. Ou *et al.*, "Simultaneous analog placement and routing with current flow and current density considerations," in *Proceedings of the 50th Annual Design Automation Conference*, 2013, pp. 1–6.

[9] R. Martins *et al.*, "Current-flow and current-density-aware multi-objective optimization of analog ic placement," *Integration*, vol. 55, pp. 295–306, 2016.

[10] B. Xu *et al.*, "Device layer-aware analytical placement for analog circuits," in *Proceedings of the 2019 International Symposium on Physical Design*, 2019, pp. 19–26.

[11] K.-H. Ho *et al.*, "Coupling-aware length-ratio-matching routing for capacitor arrays in analog integrated circuits," in *Proceedings of the 50th Annual Design Automation Conference*, 2013, pp. 1–6.

[12] Y. Li *et al.*, "Exploring a machine learning approach to performance driven analog ic placement," in *2020 IEEE computer society annual symposium on VLSI (ISVLSI)*. IEEE, 2020, pp. 24–29.

[13] Y. Li *et al.*, "A customized graph neural network model for guiding analog ic placement," in *Proceedings of the 39th International Conference on Computer-Aided Design*, 2020, pp. 1–9.

[14] A. Gusmão *et al.*, "Semi-supervised artificial neural networks towards analog ic placement recommender," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020, pp. 1–5.

[15] B. Xu *et al.*, "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)*. Westminster, CO, USA: IEEE, Nov. 2019, pp. 1–8.

[16] H. Chen *et al.*, "MAGICAL 1.0: An Open-Source Fully-Automated AMS Layout Synthesis Framework Verified With a 40-nm 1GS/s ΣΔ ADC," in *2021 IEEE Custom Integrated Circuits Conference (CICC)*. Austin, TX, USA: IEEE, Apr. 2021, pp. 1–2.

[17] H. Chen *et al.*, "MAGICAL: An Open- Source Fully Automated Analog IC Layout System from Netlist to GDSII," *IEEE Design & Test*, vol. 38, pp. 19–26, 2021.

[18] K. Kunal *et al.*, "ALIGN: Open-Source Analog Layout Automation from the Ground Up," in *Proceedings of the 56th Annual Design Automation Conference 2019*. Las Vegas NV USA: ACM, Jun. 2019, pp. 1–4.

[19] X. Gao *et al.*, "Interactive analog layout editing with instant placement legalization," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 1249–1254.

[20] X. Gao *et al.*, "Interactive Analog Layout Editing With Instant Placement and Routing Legalization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, pp. 698–711, Mar. 2023.

[21] Y. Lin *et al.*, "Intelligent and interactive analog layout design automation," in *2022 IEEE 16th International Conference on Solid-State & Integrated Circuit Technology (ICSICT)*, 2022, pp. 1–4.

[22] Y. Yin *et al.*, "Ado-llm: Analog design bayesian optimization with in-context learning of large language models," *arXiv preprint arXiv:2406.18770*, 2024.

[23] Z. Wang *et al.*, "Learn-by-compare: Analog performance prediction using contrastive regression with design knowledge," *2024 IEEE/ACM Design Automation Conference (DAC)*, 2024.

[24] G. Chen *et al.*, "Llm-enhanced bayesian optimization for efficient analog layout constraint generation," *arXiv preprint arXiv:2406.05250*, 2024.

[25] S. Poddar *et al.*, "Insight: Universal neural simulator for analog circuits harnessing autoregressive transformers," *arXiv preprint arXiv:2407.07346*, 2024.

[26] C. Liu *et al.*, "Ladac: Large language model-driven auto-designer for analog circuits," *Authorea Preprints*, 2024.

[27] Y. Lai *et al.*, "Analogcoder: Analog circuit design via training-free code generation," *arXiv preprint arXiv:2405.14918*, 2024.

[28] J. Chaudhuri *et al.*, "Spiced: Syntactical bug and trojan pattern identification in a/ms circuits using llm-enhanced detection," *arXiv preprint arXiv:2408.16018*, 2024.

[29] A. Hammoud *et al.*, "Human language to analog layout using glayout layout automation framework," in *Proceedings of the 2024 ACM/IEEE International Symposium on Machine Learning for CAD*, 2024, pp. 1–7.

[30] Z. He *et al.*, "ChatEDA: A Large Language Model Powered Autonomous Agent for EDA," Mar. 2024, arXiv:2308.10204 [cs].

[31] Z. Wang *et al.*, "Chatpattern: Layout pattern customization via natural language," https://arxiv.org/abs/2403.15434, 2024.

[32] C. Liu *et al.*, "Ampagent: An llm-based multi-agent system for multi-stage amplifier schematic design from literature for process and performance porting," *arXiv preprint arXiv:2409.14739*, 2024.

[33] B. Liu *et al.*, "Layoutcopilot: An llm-powered multi-agent collaborative framework for interactive analog layout design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2025.

[34] X. Gao *et al.*, "Joint placement optimization for hierarchical analog/mixed-signal circuits," in *2025 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE.

[35] H. Zhang *et al.*, "Sageroute: Synergistic analog routing considering geometric and electrical constraints with manual design compatibility," in *DATE*, 2023, pp. 1–6.

[36] H. Zhang *et al.*, "Sageroute2.0: Hierarchical analog and mixed signal routing considering versatile routing scenarios," in *DATE*, 2024, pp. 1–6.

[37] H. Zhang *et al.*, "Multi-scenario analog and mixed-signal circuit routing with agile human interaction (extended abstract)," in *2023 International Symposium of Electronics Design Automation (ISEDA)*, 2023, pp. 63–64.

[38] T. B. Brown *et al.*, "Language models are few-shot learners," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[39] OpenAI *et al.*, "GPT-4 Technical Report," Mar. 2024, arXiv:2303.08774 [cs].

[40] Anthropic, "Introducing the next generation of claude," https://www.anthropic.com/news/claude-3-family, 2024, accessed: 2023-04-10.