Automated Op-amp Parameter Design Based on Multiple Large Language Model Agents

Team Name: 306EDAgogogo
Participant Names: Jinyi Shen, Ji Zhuang, Zihao Chen
Supervisor Names: Fan Yang, Xuan Zeng
Affiliation: Fudan University

***Abstract*—Large language models (LLMs) show promise for analog circuit design automation, yet operational amplifier (op-amp) parameter optimization remains challenging due to domain-specific expertise requirements, precise numerical calculations, and multi-metric trade-offs. Existing LLM-based approaches focus on topology generation, leaving parameter design manual. In this paper, we propose a multi-agent LLM framework to solve the above challenges, which integrates domain knowledge, modular tools, and collaborative optimization. Tailored prompts enable in-context learning for parameter generation, while dynamic tool invocation ensures precision. Specialized agents iteratively balance conflicting metrics. Experimental results demonstrate that the proposed framework based on the Qwen2.5-7B model, without fine-tuning, achieves fully automated op-amp design by leveraging its inherent structured output and tool invocation capabilities, significantly improving efficiency and performance in the complex op-amp parameter design task.**

***Index Terms*—Large Language Model, Design Automation, Parameter Design, Operational Amplifier**

I. Introduction

Large language models (LLMs), represented by the Qwen2.5 series [1] and GPT series [2]–[4], have recently demonstrated remarkable capabilities in learning and applying human knowledge encoded in text. With billions of trainable parameters and the multi-head self-attention mechanism [5], LLMs are proficient at capturing long-range dependencies and generating contextually appropriate responses. Specifically, the Qwen2.5-7B [1] model has made significant improvements in instruction following, generating long texts, understanding structured data and generating structured outputs, particularly in JSON format.

As LLM application areas continue to expand, their potential in analog circuit design automation is gradually emerging. Operational amplifiers (op-amps), as one of the most important analog circuit modules, have to be tailored frequently to meet different design requirements. There are already several works using LLMs for the topology generation of op-amps [6]–[8]. However, op-amp parameter optimization remains a manual and time-consuming process, creating an urgent need for automated parameter design.

Despite the potential of LLMs to transform design automation, the design process of op-amp is more complex than that of highly standardized digital circuits due to the scarcity of analog circuit data, the knowledge-heavy design process, and the strongly nonlinear circuit behavior. As a result, LLMs are still in the early stages of exploration for op-amp design. Specifically, there are several challenges:

1) LLMs lack domain-specific knowledge, such as understanding the roles of circuit components, the ranges of circuit parameters and their impact on performance, leading to design failure [9].
2) Op-amp performance is highly sensitive to the values of parameters, which requires precise calculations that LLMs cannot inherently perform.
3) The design process involves complex trade-offs [10] between various performance metrics such as gain, bandwidth, and power consumption, which requires iterative optimization to achieve the optimal trade-off.

To address these challenges, we propose an op-amp parameter design framework which integrates multiple LLM agents with domain-specific knowledge and computational tools. Our contributions are summarized as follows.

1) For lack of circuit knowledge, we encode op-amp design principles into customized prompts, allowing LLMs to leverage in-context learning for parameter design.
2) For precise parameter calculations, we develop modular computational tools that LLMs can invoke during the design process.
3) For handling complex trade-offs, we introduce a multi-agent LLM-based parameter design framework, where specialized agents collaborate to iteratively optimize performance metrics.

Our approach automates op-amp design, significantly reducing the time and expertise required while achieving high-performance results. We choose the Qwen2.5-7B model as the core model in the agent framework. To maintain the model's general capabilities, such as structured output, tool invocation, and good instruction-following ability, we ultimately decide not to fine-tune the model.

The remainder of this technical report is organized as follows. Section III proposes our methods. Section IV presents the experimental results. Section V provides our conclusion.

II. Background

A. LLM Agent and Analog Circuit Design

Multiple LLMs, typically state-of-the-art general-purpose LLMs such as the Qwen2.5 series [1] and GPT series [2]–[4], are often organized as multiple *agents* [11] to collaboratively solve complex problems. Heterogeneous agents with specialized roles interact through structured communication protocols.

Recent advancements have demonstrated the adaptability of such *multi-agent* frameworks to analog circuit design. Numerous studies have focused on developing specialized agents for distinct design stages in analog circuits, such as topology design [6], [12], parameter calculation [6], and layout optimization [13], with each task allocated to dedicated agents, mirroring human design teams. These tasks are further decomposed into sub-tasks, forming an iterative optimization framework of multi-agent arrays to enhance task success rates.

B. LLM Reasoning and Analog Circuit Design

Recent advancements in LLM reasoning have enhanced the ability of a single LLM agent to tackle complex tasks by structuring their problem-solving processes. Methods such as chain of thoughts (CoT) [14], tree of thoughts (ToT) [15], and graph of thoughts (GoT) [16] enable models to reason step-by-step, explore multiple decision paths, and evaluate interconnected possibilities, respectively. These techniques guide LLMs to solve problems systematically, improving logical consistency and adaptability in scenarios requiring intricate analysis.

For example, reference [6] models the complex parameter calculation process as acyclic CoT graphs to enforce deterministic constraint propagation, while [13] formalizes the iterative layout tuning workflow as feedback-rich GoT structures that dynamically incorporate parasitic extraction results and design rule checks.

C. Retrieval-augmented Generation and Analog Circuit Design

Retrieval-augmented Generation (RAG) [17] enhances LLMs by dynamically grounding responses in external knowledge through a two-stage process: a retriever semantically searches databases to identify relevant evidence, while a generator synthesizes outputs conditioned on both retrieved content and parametric knowledge. Therefore, RAG can integrate precise, context-relevant information into the LLM's reasoning process.

In analog circuit design, numerous complex knowledge components, such as the suitability matching between different circuit architectures and design requirements [8], specific netlists of various circuits [7], and empirical rules for parameter and layout tuning [13], [18], are often manually summarized and refined in knowledge bases for on-demand retrieval by agents.

III. Methods

In this section, we propose the automated op-amp parameter design framework based on multiple LLM agents. Section III-A first describes the op-amp structure selection and reasons. Section III-B proposes the LLM-based op-amp parameter design framework, where all agents are detailed in Section III-C, and the knowledge base is presented in Section III-D.

A. Op-amp Structure Selection

The design problem allows for the pre-specification of the op-amp circuit topology. We select the telescopic cascode op-amp as the circuit structure, as illustrated in Fig. 1, where the module in the red box is the amplifier circuit, while the module in blue box is the bias circuit. The telescopic cascode op-amp is chosen due to the following reasons:

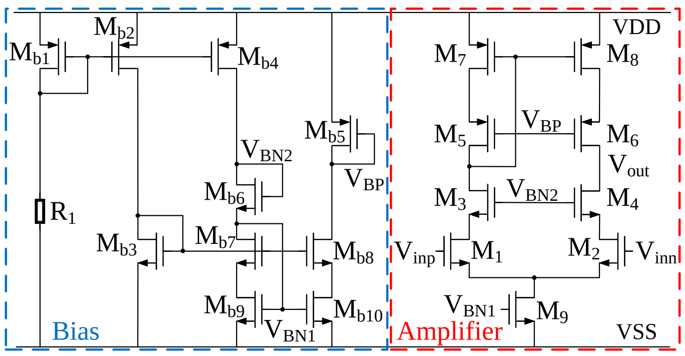


Fig. 1: The structure of a telescopic cascode op-amp.

1) The telescopic cascode amplifier employs a single amplification stage, which offers lower power consumption compared to two-stage or three-stage amplifiers. Since no frequency compensation is required, it also achieves higher gain-bandwidth product (GBW), larger phase margin (PM), and higher slew rate (SR).
2) Although the gain of single-stage amplifiers is typically lower than that of two-stage or multi-stage amplifiers, the cascode structure can enhance the gain on the other hand. The active load is implemented as a low-voltage current mirror, which not only achieves single-ended output, but also consumes less voltage headroom compared to conventional cascode current mirrors [10].

In the amplifier module, the transistor M_9 serves as the tail current source, while M_1 and M_2 constitute the input differential pair. Transistors M_3 and M_4 function as cascode devices, and M_5–M_8 form a low-voltage current mirror.

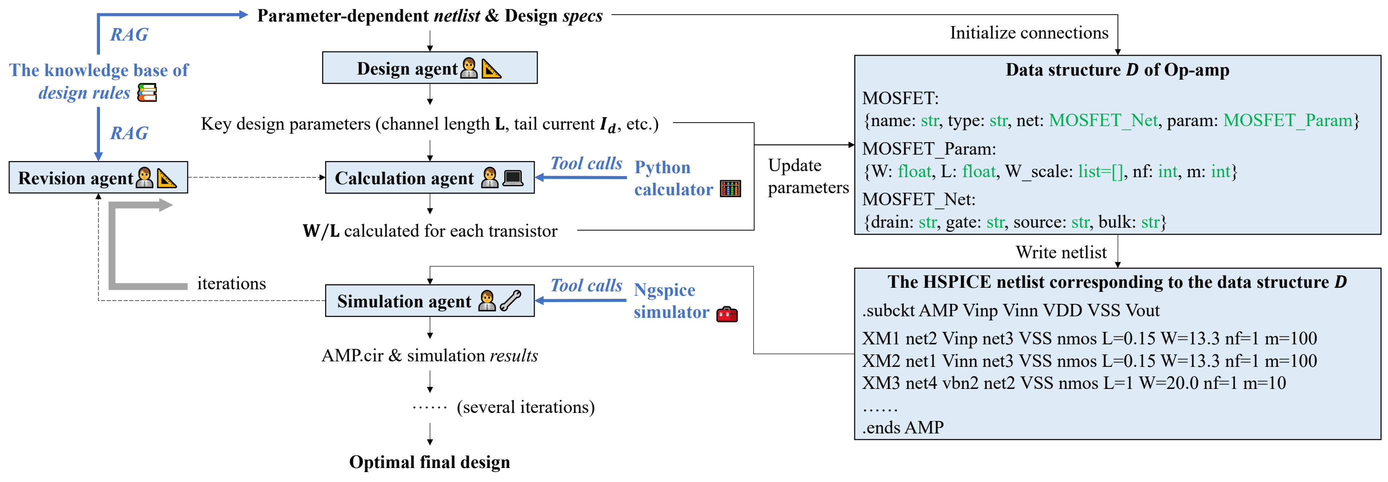
In the bias module, the resistor R_1 generates the base current. Diode-connected transistors M_{b6} and M_{b5} produce the required bias voltages V_{BN2} and V_{BP} for the op-amp, respectively. The bias voltage V_{BN1} is derived from the diode-connected transistor M_{b3} and the low-voltage current mirror formed by M_{b7}–M_{b10}. The bias circuit is designed to ensure that the generated voltages precisely match the op-amp's biasing requirements.

The design variables include the widths (W) and lengths (L) of all transistors in the op-amp (M_1–M_9), all transistors in the bias circuit (M_{b1}–M_{b10}), and the value of resistor R_1.

B. Multi-agent Parameter Design Framework

Op-amp design is a complex process that involves multiple distinct steps. Additionally, these steps must be performed sequentially, with multiple design iterations and potential backtracking until the final design meets the specified requirements. The diversity of tasks, heterogeneous capability needs, and potential backtracking in multiple design iterations pose significant challenges for using a single LLM agent to complete the entire design process.

The mainstream approaches for solving complex problems require multi-step reasoning: one approach is to enable LLMs to possess inherent CoTs [14] and GoTs [16], typically through supervised fine-tuning [19] or reinforcement learning [20]; the other approach involves breaking the complex problem into


Parameter-dependent netlist & Design specs
RAG
The knowledge base of design rules
RAG
Design agent
Key design parameters (channel length L, tail current I_d, etc.)
Revision agent
Calculation agent
Tool calls
Python calculator
W/L calculated for each transistor
iterations
Simulation agent
Tool calls
Ngspice simulator
AMP.cir & simulation results
······ (several iterations)
Optimal final design
Initialize connections
Update parameters
Data structure D of Op-amp
MOSFET:
{name: str, type: str, net: MOSFET_Net, param: MOSFET_Param}
MOSFET_Param:
{W: float, L: float, W_scale: list=[], nf: int, m: int}
MOSFET_Net:
{drain: str, gate: str, source: str, bulk: str}
Write netlist
The HSPICE netlist corresponding to the data structure D

```
.subckt AMP Vinp Vinn VDD VSS Vout
XM1 net2 Vinp net3 VSS nmos L=0.15 W=13.3 nf=1 m=100
XM2 net1 Vinn net3 VSS nmos L=0.15 W=13.3 nf=1 m=100
XM3 net4 vbn2 net2 VSS nmos L=1 W=20.0 nf=1 m=10
......
.ends AMP
```

Fig. 2: The proposed op-amp parameter design framework. The left half is the multi-agent parameter design framework, while the right half illustrates our data structure. The data structure consists of two parts: the netlist structure and the parameters, initialized by the input specs and updated synchronously when agents modify parameters. During simulation, the data structure is parsed into the actual netlist for the simulation agent.

several steps, using a multi-agent framework to decompose different tasks into separate agents and enable collaboration among them. This framework is often customized on the basis of human experience [11].

We adopt the second approach in this project, where we develop a customized multi-agent LLM-based parameter design framework, as shown in Fig. 2. These steps are executed in a specified order, with each step performed by a dedicated LLM agent. The process begins with the design agent, which determines the key design parameters based on the given topology. Next, the calculation agent calculates the values of the remaining parameters in the netlist based on the values of key design parameters. Subsequently, the simulation agent outputs the netlist stored in the program and invokes the *ngspice* simulator for simulation. When the simulation is finished, the revision agent compares the simulated performance metrics against the target specifications and modifies the design parameters according to a set of design rules carefully crafted from human expertise. Finally, a top-level framework orchestrates the execution sequence of the agents, supports multi-round iterative optimization, and continuously refines the circuit performance to find the optimal design within a limited number of iterations.

C. Detailed Agent Execution

The detailed execution processes of different agents in our framework is presented as follows:

1) **Design agent**: This agent parses the topology netlist and design specifications provided in the prompt. It initializes key design parameters (e.g., tail current, channel length of input and load transistors) based on the parameter design rules and design goals. For example, op-amp input transistors use short channel lengths to maximize speed and transconductance while minimizing noise, whereas load transistors employ medium lengths to optimize output resistance, matching, and headroom, balancing trade-offs between bandwidth and gain.
2) **Calculation agent**: This agent handles all computational tasks in the parameter design process. It calculates the width of transistors to satisfy the maximum transistor area. It also calculates the resistance of R_1 using the Ohm's law. All the required calculations are pre-written in this agent and are available as individual tools for the agent to invoke. It leverages multi-threading for parallel computation of design parameters.
3) **Simulation agent**: This agent first writes all the computed design parameters stored in the program into the circuit netlist file, and then utilizes multi-threading to execute AC, DC, and transient simulations in parallel using `ngspice`. These simulations are performed only under the TT process corner to save time. It then reads the performance results and stores the circuit parameters and current performance metrics.
4) **Revision agent**: This agent begins by comparing the current simulated performance metrics against the target specifications to identify any unmet requirements. Next, it retrieves a set of corresponding parameter adjustment rules carefully crafted from human expertise. It then applies these rules to modify the circuit parameters iteratively, thus optimizing the overall performance.

D. Circuit Parameter Design Knowledge Base

Our circuit parameter design knowledge base consists of three main components:

1) **Circuit structure and functional description**: The circuit structure involves identifying key components—input transistors, load transistors, and bias transistors—based on their interconnections. The input transistors convert differential input signals into current, necessitating high transconductance. The load transistors

then reconvert this current into voltage, requiring high intrinsic gain. Finally, the bias transistors generate the necessary reference voltages, and their channel lengths should match those of the corresponding transistors in the op-amp to ensure proper biasing.

2) **Initial parameter design rules**: These rules define the allowable ranges for L and W based on the fabrication process, as well as the tail current range derived from slew rate and current requirements.
3) **Parameter adjustment rules**: These rules guide the modification of circuit parameters when performance metrics fail to meet design requirements. For example, if the circuit gain is insufficient, the transistor channel length is increased to enhance intrinsic gain. If the PM is inadequate, the maximum transistor area is reduced to minimize the impact of parasitics on circuit performance.

The circuit parameter design knowledge base is provided to the respective agents in the form of prompts as part of the agent input. Specifically, the circuit structure and functional description and initial parameter design rules are given to the design agent, while the parameter adjustment rules are provided to the revision agent. Our knowledge base comprises a total of 357 tokens, making it highly compact and lightweight.

E. Pseudo-code for parameter design framework

We present the pseudo-code of our multi-agent framework for op-amp parameter tuning in Algorithm 1. First, our specialized agents (design, calculation, simulation, and revision agents) with their respective functional prompts are initialized. In the first iteration, the data structure $\mathcal{D}$ is initialized, and key design variables are determined with design agent. During each iteration, the calculation agent first determines transistor dimensions (W and L values) based on current design parameters. The simulation agent then validates these parameters against performance constraints through circuit simulation. If all specifications are satisfied, the algorithm terminates early and returns the optimized design. When constraints are violated, the revision agent systematically updates the design data structure $\mathcal{D}$ to refine the parameters. This cyclic process continues until either successful constraint satisfaction occurs or the maximum iteration count T is reached. The architecture enables collaborative optimization where each agent contributes distinct capabilities: initial design generation, parameter calculation, performance verification, and iterative improvement, respectively. The data structure $\mathcal{D}$ serves as both design memory and communication medium between agents throughout the optimization sequence.

Algorithm 1: Multi-agent framework for op-amp parameter tuning

```
input : System prompt, Prompts for agents, Iterations $T$
1  Initialize data structure $\mathcal{D}$, design agent, calculation
     agent, simulation agent, revision agent;
2  Parse the op-amp netlist in the system prompt and
     update component connections in $\mathcal{D}$;
3  Initialize key design variables with design agent;
4  for $t = 1$ to $T$ do
5      Get W and L for all transistors with calculation
         agent;
6      Update component parameters in $\mathcal{D}$;
7      Get simulation results with simulation agent;
8      if Simulation results meet all constraints then
9          break;
10     Update component parameters in $\mathcal{D}$ with revision
         agent;
11 return the current best topology $G^*$
```

IV. Experimental Results

A. Experimental Settings

Our multi-agent LLM-based framework is deployed on the cloud computing platform provided by the organizing committee, which is based on *Phytium Tian 710* CPU processor with ARMv9 architecture.

The parameter design for the telescopic cascode op-amp uses the proposed framework under the `skywater130` PDK. The PDK allows for a range of 0.15 μm to 2 μm for transistor length, and 0.35 μm to 100 μm for transistor width. The op-amp operates under a supply voltage of 1.8 V and drives a load capacitor of 2 pF. The op-amps are simulated using `ngspice43`.

The op-amp performance constraints under the TT corner are as follows.

$$\begin{array}{lcl} \text{Gain} & > & 60\text{ dB}, \\ \text{PM} & > & 60^\circ, \\ \text{GBW} & > & 20\text{ MHz}, \\ \text{I}_{\text{dc}} & < & 500\ \mu\text{A}, \\ \text{SR} & > & 20\text{ V}/\mu\text{s}. \end{array} \tag{1}$$

Our framework in implemented using Python 3.10 and the agents are designed using the ollama-python library. All the LLMs in our framework are based on the Qwen2.5-7B model, and we use the default settings of the ollama parameters related to model invocation. We set the maximum number of iterations to 5 to achieve a good balance between op-amp performance and runtime. Considering the inherent randomness of LLM responses, we repeat the experiments for ten times.

B. Design Process Demonstration

Due to space limitations, we only present the design process for the op-amp with the best performance. The design process involve four iterations that are detailed below.

Iteration 1: The design agent initially sets the input transistor length to 0.15 μm to improve the transconductance and other transistor lengths to 1 μm to balance the intrinsic gain and transistor area for load and bias transistors. Transistor areas are constrained to a maximum of 200 μm^2, with widths determined by the calculation agent. The simulation agent evaluates the initial design and extracts the performance metrics.

Iteration 2: Upon comparison with the target specifications, the revision agent identifies a failure in gain and PM, and adjusts the design by increasing the input transistor length to 0.25 μm to improve gain and halving the maximum transistor area to 100 μm^2 to enhance PM. The calculation agent then recomputes all transistor widths.

Iteration 3: Despite the above modification, the revised design still fails to meet the gain and PM requirements. The revision agent further increases the input transistor length to 0.35 μm and reduces the maximum transistor area to 50 μm^2.

Iteration 4: While this iteration satisfies the gain requirement, the PM remains insufficient. Given its proximity to the target PM, the revision agent proposes a final adjustment, reducing the maximum transistor area to 45 μm^2. The resulting design successfully meets all requirements under the TT corner. To minimize optimization time, no further refinements are performed under other corners.

The complete design process requires 128.39 seconds, demonstrating a 2.70× to 19.73× speedup compared to all other valid submissions (excluding implausibly short runtime that violate physical constraints). As shown in Fig. 3, the runtime is dominated by the calculation and revision agents, which account for the majority of the computational effort due to their iterative invocations and the computational overhead of their design optimization tools.

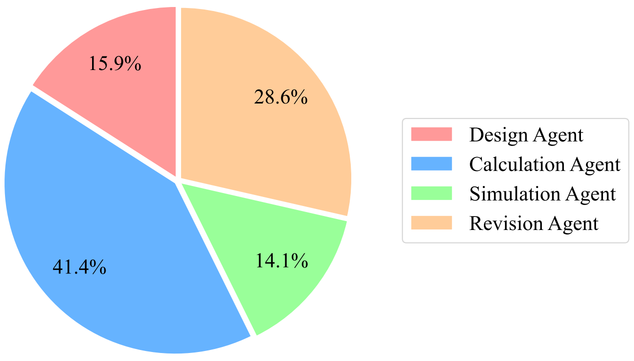


Fig. 3: Op-amp design time breakdown.

C. *Op-amp Performance*

The final transistor parameter values for the op-amp are summarized in Table I. We conduct DC, AC, and transient simulations across five process corners to evaluate the op-amp's performance. As shown in Table II, all performance metrics—except for gain—satisfy the specified constraints under varying corners, demonstrating the robustness of our design. Fig. 4 shows the gain and phase frequency responses of the op-amp under the TT process corner, while Fig. 5 demonstrates its transient response to a square-wave input. These measurement results confirm the proper functionality of the designed op-amp.

Table III presents the worst-case performance for the five key metrics, along with three critical op-amp parameters: common-mode rejection ratio (CMRR), power supply rejection ratio (PSRR), and noise. Compared to other submitted designs, our

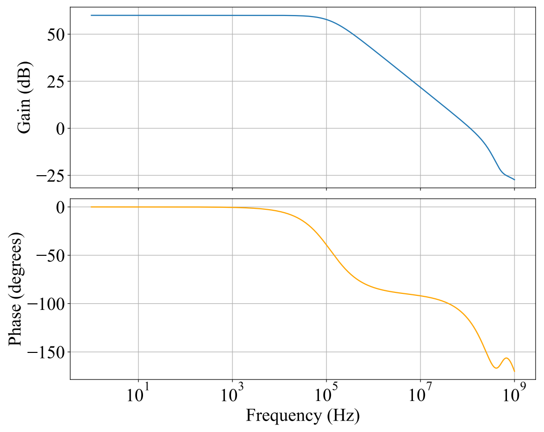


Fig. 4: Op-amp Design Time Distribution.

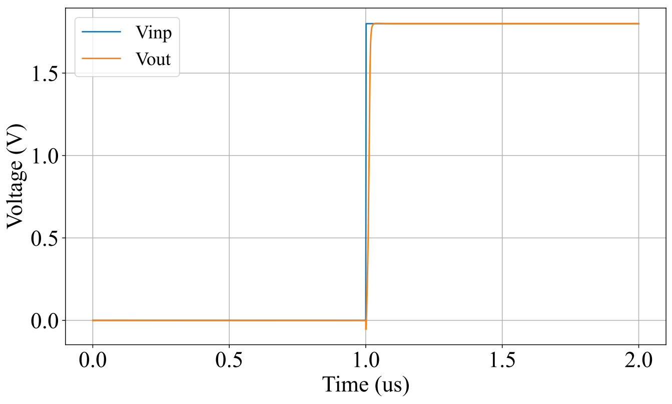


Fig. 5: Op-amp Design Time Distribution.

op-amp achieves the highest GBW, SR, and CMRR, resulting in the highest overall circuit performance score.

TABLE I: Op-amp Parameter Values

Parameter Name	Parameter Value
$W_{1,2}$ / $L_{1,2}$	12.86 μm × 10 / 0.35 μm
$W_{3,4}$ / $L_{3,4}$	45 μm / 1 μm
$W_{5,6}$ / $L_{5,6}$	45 μm / 1 μm
$W_{7,8}$ / $L_{7,8}$	45 μm / 1 μm
W_9 / L_9	45 μm / 1 μm
$W_{b1,b2,b4}$ / $L_{b1,b2,b4}$	45 μm / 1 μm
W_{b3} / L_{b3}	0.42 μm / 1 μm
W_{b5} / L_{b5}	0.42 μm / 1 μm
W_{b6} / L_{b6}	4.20 μm / 1 μm
$W_{b7,b8}$ / $L_{b7,b8}$	4.20 μm / 1 μm
$W_{b9,b10}$ / $L_{b9,b10}$	4.20 μm / 1 μm
R_1	127.94 kΩ

V. CONCLUSION

This paper introduces a multi-agent LLM-based parameter design framework, with a circuit parameter design knowledge base carefully integrated into the prompts. This framework is capable of automatically designing and calculating parameters,

TABLE II: Op-amp Performance under Different Corners

Corner	TT	FF	SF	FS	SS
Gain (dB)	60.01	52.98	66.94	48.43	64.39
GBW (MHz)	114.42	120.58	110.81	108.57	103.95
PM (deg)	61.22	60.38	62.94	61.62	62.59
Idc (μA)	182.76	201.30	165.94	197.56	165.11
SR (V/μs)	123.33	135.87	117.47	126.68	109.32

TABLE III: Op-amp Performance Summary

Performance Metric	Worst Corner	Value
Gain	FS	48.43 dB
GBW	SS	103.95 MHz
PM	FF	60.38 deg
I_{dc}	FF	201.30 μA
SR	SS	109.32 V/μs
CMRR	TT	110.25 dB
PSRR	TT	69.01 dB
Noise	TT	0.17 μV/$\sqrt{\text{Hz}}$

simulating circuit netlists, and performing rule-based iterative optimization for op-amps with a given circuit structure.

In reviewing this work, there are some advantages of using such a multi-agent framework for parameter design:

1) It allows LLMs to automatically determine the required design parameters directly from natural language specifications.
2) It demonstrates substantially higher design efficiency compared to traditional manual parameter design.
3) It can automatically automatically execute computational functions for precise parameter determination and invoke simulation tools for performance verification.
4) By incorporating human expert knowledge during iterative optimization, it ensures an interpretable design process while optimally resolving complex performance trade-offs.

We plan to investigate the integration of LLMs into analog circuit design automation through two key research directions:

1) **End-to-end design automation**: We will develop methods to incorporate the complete op-amp design workflow—from topology selection through parameter optimization to final layout generation—into our multi-agent framework. This will leverage both RAG and fine-tuning approaches. Furthermore, we will investigate synergistic combinations of traditional optimization algorithms with LLM-guided parameter tuning to further enhance design efficiency.
2) **Autonomous design reasoning**: While current implementations rely on human expertise to decompose design tasks, we aim to develop LLM capabilities for autonomous design decomposition. This involves training models to intelligently partition analog circuit design problems into optimal procedural sequences to replace manual, rule-based optimization methods.

ACKNOWLEDGEMENTS

This project is an entry in the IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS) 2025 Grand Challenge. The configuration of the circuit environment is supported by the committee of Track 3, and the LLM deployment is supported by Alibaba Cloud's Tianchi platform.

REFERENCES

[1] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei *et al.*, "Qwen2.5 technical report," *arXiv preprint arXiv:2412.15115*, 2024.

[2] L. Floridi and M. Chiriatti, "Gpt-3: Its nature, scope, limits, and consequences," *Minds and Machines*, vol. 30, pp. 681–694, 2020.

[3] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[4] R. Islam and O. M. Moushi, "Gpt-4o: The cutting-edge advancement in multimodal llm," *Authorea Preprints*, 2024.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[6] Z. Chen, J. Huang, Y. Liu, F. Yang, L. Shang, D. Zhou, and X. Zeng, "Artisan: Automated operational amplifier design via domain-specific large language model," *2024 61th ACM/IEEE Design Automation Conference (DAC)*, 2024.

[7] Y. Lai, S. Lee, G. Chen, S. Poddar, M. Hu, D. Z. Pan, and P. Luo, "Analogcoder: Analog circuit design via training-free code generation," *arXiv preprint arXiv:2405.14918*, 2024.

[8] C. Liu, W. Chen, A. Peng, Y. Du, L. Du, and J. Yang, "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.

[9] K. Chen, Y. Yang, B. Chen, J. A. H. López, G. Mussbacher, and D. Varró, "Automated domain modeling with large language models: A comparative study," in *2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 2023, pp. 162–172.

[10] B. Razavi, *Design of Analog CMOS Integrated Circuits*, ser. Electrical Engineering Series. McGraw-Hill, 2001.

[11] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *Ieee Access*, vol. 6, pp. 28 573–28 593, 2018.

[12] C.-C. Chang, Y. Shen, S. Fan, J. Li, S. Zhang, N. Cao, Y. Chen, and X. Zhang, "Lamagic: Language-model-based topology generation for analog integrated circuits," *arXiv preprint arXiv:2407.18269*, 2024.

[13] B. Liu, H. Zhang, X. Gao, Z. Kong, X. Tang, Y. Lin, R. Wang, and R. Huang, "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.

[14] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[15] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," 2023.

[16] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk *et al.*, "Graph of thoughts: Solving elaborate problems with large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 16, 2024, pp. 17 682–17 690.

[17] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.

[18] D. V. Kochar, H. Wang, A. Chandrakasan, and X. Zhang, "Ledro: Llm-enhanced design space reduction and optimization for analog circuits," *arXiv preprint arXiv:2411.12930*, 2024.

[19] N. Ho, L. Schmid, and S.-Y. Yun, "Large language models are reasoning teachers," *arXiv preprint arXiv:2212.10071*, 2022.

[20] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.