

MenTeR: A fully-automated Multi-agent workflow for end-to-end RF/Analog Circuits Netlist Design

Pin-Han Chen
MediaTek Technology

West Lafayette, Indiana
peter-ph.chen@mediatek.com

Yu-Sheng Lin
MediaTek

Hsinchu, Taiwan
bob-ys.lin@mediatek.com

Wei-Cheng Lee
MediaTek

Hsinchu, Taiwan
ds_wei-cheng.lee@mediatek.com

Tin-Yu Leu
MediaTek

Hsinchu, Taiwan
magnus.leu@mediatek.com

Po-Hsiang Hsu
MediaTek

Hsinchu, Taiwan
stanley-ph.hsu@mediatek.com

Anjana Dissanayake
MediaTek Technology

West Lafayette, Indiana
anjana.dissanayake@mediatek.com

Sungjin Oh
MediaTek Technology

West Lafayette, Indiana
sungjin.oh@mediatek.com

Chinq-Shiun Chiu
MediaTek USA

West Lafayette, Indiana
cs.chiu@mediatek.com

Abstract—RF/Analog design is essential for bridging digital technologies with real-world signals, ensuring the functionality and reliability of a wide range of electronic systems. However, analog design procedures are often intricate, time-consuming and reliant on expert intuition, and hinder the time and cost efficiency of circuit development. To overcome the limitations of the manual circuit design, we introduce MenTeR – a multi-agent workflow integrated into an end-to-end analog design framework. By employing multiple specialized AI agents that collaboratively address different aspects of the design process, such as specification understanding, circuit optimization, and test bench validation, MenTeR reduces the dependency on frequent trial-and-error-style intervention. MenTeR not only accelerates the design cycle time but also facilitates a broader exploration of the design space, demonstrating robust capabilities in handling real-world analog systems. We believe that MenTeR lays the groundwork for future “RF/Analog Copilots” that can collaborate seamlessly with human designers.

Index Terms—RF/Analog Design, Large Language Models (LLMs), Multi-Agent, Schematic Design, Chain-of-Stage (CoS), Diagram-Aware Retrieval-Augmented Generation (DA-RAG)

I. INTRODUCTION

The recent progress of Large Language Models (LLMs) has led to an increasing numbers of LLM applications in scientific and engineering fields such as mathematical reasoning, pharmaceutical development, and chip design. For instance, in the field of digital circuit design, Liu et al. [1] introduced the first domain-adapted LLM, which demonstrated the potential of using legacy chip design documents to increase the design capabilities of LLM. BlockLove et al. [2] investigated the applications of LLMs in translating natural languages into Hardware Description Languages (HDL), showing the design example of an 8-bit microprocessor. In the field of RF/Analog circuit design, Lai et al. [3] proposed the first training-free LLM application to automate design of elementary circuit blocks through Python code generation. Liu et al. [4] introduced another LLM-based multi-agent system that automates multi-stage amplifier schematic design by integrating literature analysis, mathematical reasoning, and device sizing agents.

While these works have established a good foundation for LLM applications in circuit design, there are still significant challenges for practical industrial deployment. For example, [3] leveraged LLMs to solve simplified analog design tasks and built a corresponding benchmark; however, there still exist several gaps between the benchmark problem sets of [3] and industrial-scale design problems. While [4] demonstrated that a multi-stage amplifier can be designed with a multi-agent system, the scalability of the approach to more complex amplifier designs or different circuit architectures is not yet explored.

To mitigate these gaps, we aim to build an AI-based solution for RF/Analog design, with the ultimate goal of assisted system-to-transistor-level implementation of complex circuits and systems such as Bandgap Reference (BGR) and Phase-Locked Loop (PLL). To achieve this goal, there are several questions that should be answered:

- What are the capabilities and limitations of current LLMs in RF/Analog circuits design? What techniques and data can we use to improve LLMs in reasoning required for practical designs?
- RF/Analog circuits design is an intricate process that relies on the designer’s intuitions and years of experience; how can we decompose this into multiple stages and integrate into an automated design flow?
- How can we ensure the scalability and reliability of LLM-based systems for practical RF/Analog circuit designs?

To address these challenges, we propose MenTeR: A fully-automated multi-agent workflow for end-to-end RF/Analog circuits netlist design, illustrated in Figure 1. MenTeR is designed to facilitate analog designers with specification reasoning, document search, testbench generation, and schematic design. Our contributions are:

- 1) To the best of our knowledge, MenTeR is the first fully-automated multi-agent framework for end-to-end RF/Analog design using LLM, which can generate analog circuit netlists of a single-block, multi-blocks, and

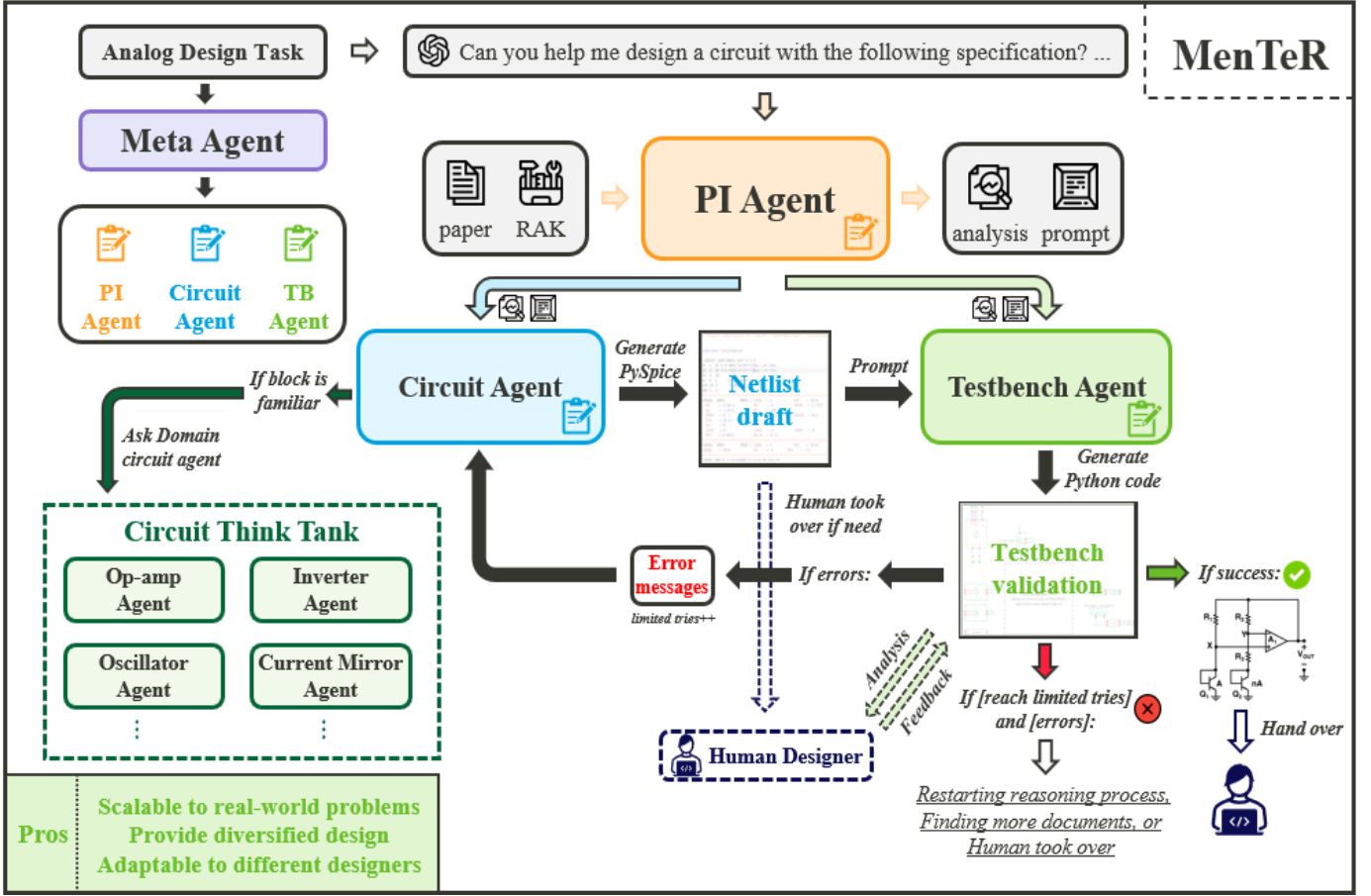


Fig. 1: **Overview of MenTeR.** A fully-automated multi-agent workflow for RF/Analog circuit design that translates specifications to netlist. MenTeR is designed with interactive checkpoints where human designers can intervene as needed.

ultimately system-level through a scalable approach;

- 2) A practical study of LLMs' reasoning capabilities enhancement with Diagram-Aware Retrieval-Augmented Generation (DA-RAG) and self-referential techniques, which also increases LLMs' capabilities in test bench writing, leading to minimal human engineer intervention;
- 3) We introduce a Chain-of-Stage (CoS) reasoning agent for RF/Analog design that integrates domain-specific knowledge with the Chain-of-Thought (CoT) technique, enhancing the reasoning capabilities of LLMs without requiring extensive training data.

II. BACKGROUND

A. Enhancing Reasoning Capabilities of LLMs

In this section, we review key techniques for enhancing LLMs' reasoning capabilities, which inform our approach to improving LLMs' RF/Analog design performance.

Prompt Engineering. Deng et al. [6] proposed a "Rephrase and Respond" (RaR) method that enables language models to rephrase input questions for better clarity, significantly improving models' performance by resolving ambiguities in human-LLM communication. Wei et al. [7] introduced Chain-of-Thought (CoT) prompting, where models were guided to generate step-by-step reasoning processes before producing

final answers, improving their performance on complex reasoning tasks. Based on this approach, Yao et al. [8] proposed Tree of Thoughts (ToT) prompting, improving performance over CoT prompting in complex reasoning tasks by allowing models to explore and evaluate multiple reasoning paths.

These prompting techniques provide efficient mechanisms for us to evaluate and enhance LLMs' capabilities in analog design, where structured reasoning and alternative solution explorations are particularly valuable.

Retrieval-Augmented Generation. Retrieval-Augmented Generation (RAG) proposed by Lewis et al. [9] has been a widely used technique to increase the ability of LLMs in knowledge-intensive language tasks. By combining a pre-trained seq2seq model with a dense vector retrieval system over text-based documents, it enables knowledge updates without model retraining. In analog design, [4] and [10] have used the basic RAG technique with analog design documents to enhance LLMs' capabilities in a complex design task.

Fine Tuning. Moreover, aligning language models with human preferences has proven to be effective in enhancing reasoning capabilities. Ouyang et al. [11] demonstrated that fine-tuning language models with Reinforcement Learning from Human Feedback (RLHF) significantly improves model alignment with user intentions. However, RLHF is a complex

and often unstable procedure that involves two main steps:

- 1) Fitting a reward model to reflect human preferences.
- 2) Fine-tuning a language model with reinforcement learning to maximize this estimated reward.

To address these challenges, Rafailov et al. [12] introduced Direct Preference Optimization (DPO), a method that bypasses explicit reward modeling and reinforcement learning by directly training language models from human preferences through a simple classification loss. Building on this approach, Meng et al. [13] proposed SimPO, a simplified preference optimization approach that improved DPO by using a reference-free reward formulation based on the average logarithmic probability of a sequence. Most recently, DeepSeek [14] demonstrated that reasoning capabilities can be significantly enhanced by large-scale reinforcement learning on high-quality Chain-of-Thought (CoT) data.

When we take a closer look at these fine-tuning methodologies, it becomes apparent that effective RF/Analog design fine-tuning requires either sophisticated reward modeling tailored to circuit designers' preferences or the development of a high-quality CoT dataset specific to the designs. Both approaches represent substantial long-term commitments. In this paper, we pay attention to the latter direction. By integrating our multi-agent workflow into daily design processes, we aim to not only solve fundamental RF/Analog circuit problems but also collect the reasoning paths employed in analog circuit design systematically. This helps us prepare the necessary steps for building a domain-adapted reasoning model to assist with more complex circuit designs.

developed AutoGen, an open-source framework that facilitated multi-agent conversations between customizable LLM-powered agents, human inputs, and tools to tackle complex tasks across mathematics, coding, and decision-making domains. Building on this concept, Song et al. [16] proposed Captain Agent, an adaptive framework that dynamically assembled and managed teams throughout the task-solving process, providing a valuable reference for formulating our multi-agent workflow. Based on these insights, we have implemented a simplified multi-agent framework and compared it with a single-agent approach, as illustrated in Figure 2.

While a single-agent approach offers implementation simplicity for analog design tasks, it inevitably encounters bottlenecks limited by the capabilities of the underlying Large Language Model. As we attempted to solve complex design challenges using LLM-based methodologies, our proposed multi-agent system demonstrated significant benefits.

In our simplified multi-agent framework, we implemented a Meta Agent serving as the central orchestrator, which analyzes incoming design tasks to determine which specialized domain experts are required. Upon receiving an analog design specification, the Meta Agent decomposes complex problems into manageable sub-tasks, delegates these components to appropriate domain specialist agents, and oversees the entire workflow.

With this simplified multi-agent approach, our method already outperformed single-agent work such as [3] (see Table I for more details). However, the structure of this framework can be further optimized to integrate into contemporary design flows. The detail of our proposed MenTeR framework will be interpreted in detail in Section III.

III. OVERVIEW OF MENTER

In this section, we introduce the proposed MenTeR framework, which decomposes an analog design task into multiple stages, including specification understanding, document search, circuit netlist design, and test bench generation. The overview of MenTeR is illustrated in Figure 1, and the flow of each agent is illustrated in Figure 3.

A. PI Agent

PI agent is designed to play the role of the "Primary Investigator," who interacts with the users, manages the workflow, and assigns tasks to corresponding agents. Given an analog design task, the PI agent should leverage the DA-RAG agent to process design requirements using multiple knowledge sources: technical papers (e.g., [18], [20]), Rapid Adoption Kit (RAK), and textbook chapters (e.g., [17]).

After retrieving the knowledge, PI agent defines the tasks and assigns them to different agents respectively. This paradigm transforms abstract user requirements into well-defined specifications, which bridges the semantic gap between human design intent and formal circuit requirements.

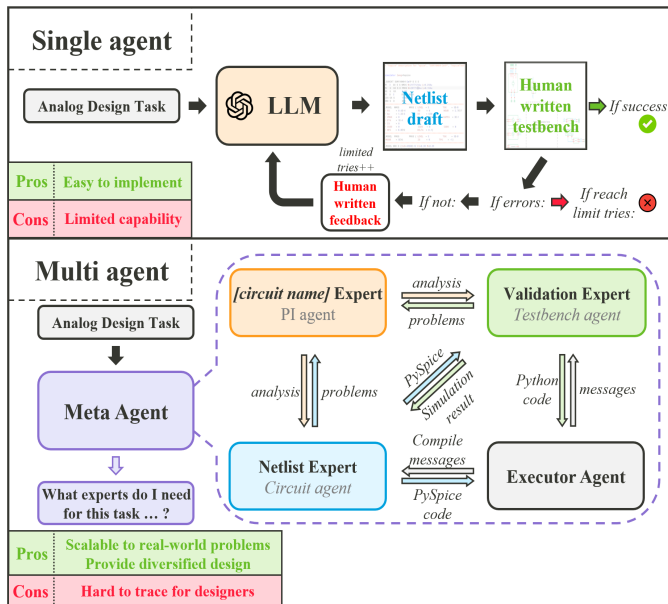


Fig. 2: Comparison between single-agent and simplified multi-agent frameworks.

B. The Necessity of a Multi-Agent Workflow

Here, we investigate the importance of a multi-agent workflow to achieve the aforementioned goal. Wu et al. [15]

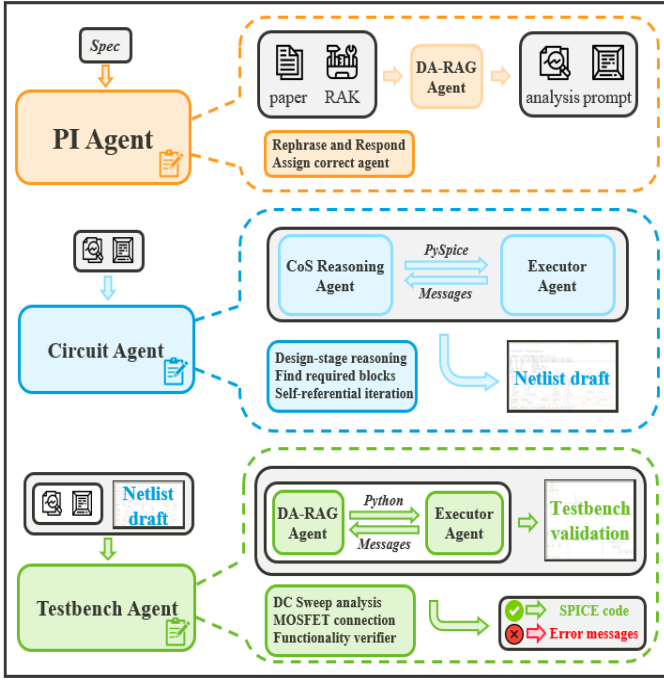


Fig. 3: Overview of MenTeR agents.

B. Circuit Agent

Upon receiving processed requirements from the PI Agent, the Circuit Agent (CA) functions as a "senior analog designer," generating the actual circuit design in PySpice¹ format. The CA's internal architecture consists of two primary components: the Chain-of-Stage (CoS) agent and the Executor agent. The CoS agent manages design stage reasoning, identifies familiar circuit blocks, and conducts self-referential iteration for design refinement.

When tasked with a design challenge, the Circuit Agent first determines the necessary design stages required for the desired circuit implementation. For familiar design blocks, it can consult domain experts in the Circuit Think Tank to leverage established design patterns. After formulating the initial design, the Circuit Agent produces a netlist draft which is then passed to the Executor agent. This secondary agent performs essential validation checks including syntax verification and other critical error detection processes. Any errors identified during this preliminary validation are routed back to the CoS agent for targeted refinement.

Once the netlist passes these basic verification checks, the validated draft is forwarded to the Testbench Agent for comprehensive simulation and testing.

C. Testbench Agent

By leveraging DA-RAG (refers to Section III-E2), the Testbench Agent (TBA) defines appropriate simulations required for the specific circuit, implementing three specialized validation components: DC sweep checker, MOSFET (and other primitive elements such as resistors) connection checker, and

functionality verifier. TBA generates corresponding Python code to validate the PySpice implementation. In the future, more advanced check agents such as steady-state large-signal operating point check can be included to facilitate system-level design.

Similar to the Circuit Agent's workflow, the testbench code is passed to an Executor agent to verify syntax correctness and ensure proper simulation. Upon completion of the validation process, it either confirms successful design verification or produces detailed error messages that are routed back to the Circuit Agent for targeted design refinement.

D. Circuit Think Tank (CTT)

As we deploy MenTeR in real-world applications, we anticipate encountering increasingly diverse circuits that share fundamental design concepts. To leverage these similarities and reduce complexity, we established the Circuit Think Tank—a knowledge repository of specialized circuit expertise.

After a circuit design is successfully completed and validated by the Testbench Agent, we capture its critical attributes including circuit name, specifications, reasoning stages, netlist implementation, and relevant interaction history. This repository serves dual purposes: it facilitates accelerated reasoning for new analog design tasks by providing access to established design patterns, and it constitutes a valuable dataset for future domain-adapted fine-tuning of our agents.

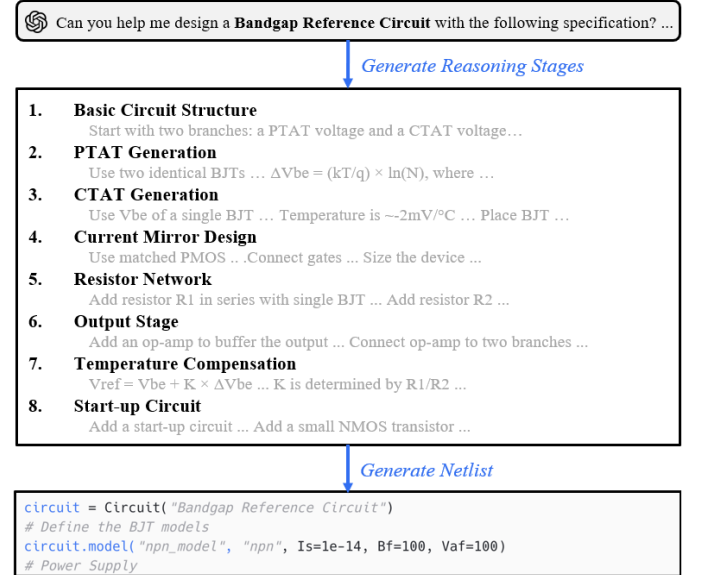


Fig. 4: CoS reasoning stages for the BGR circuit.

E. Sub-Agents

Moreover, we construct several sub-agents to equip techniques for each agent, including:

1) **CoS Reasoning Agent:** In contrast to conventional one-pass approaches, we propose a CoS reasoning agent to decompose the RF/Analog design problem into a sequence of specialized sub-tasks, or "stages." Figure 4 illustrates this methodology through the reasoning stages developed for a

¹<https://github.com/PySpice-org/PySpice>

Bandgap Reference circuit. By treating each stage as an independent task that feeds its intermediate outputs into the subsequent stage, CoS ensures that critical design stages and constraints are addressed during the generation process.

For example, the first stage of our CoS reasoning agent leverages task-relevant information (e.g. from textbooks or prior designs know-how from agents in CTT) to establish the fundamental design requirements. The system then transitions into a parameter synthesis stage, where CoT will be used to follow this hierarchical system to propose initial component values. Specifically, each stage stores both its prompts and outputs as prior knowledge, allowing for iterative updates and cross-referencing between stages.

2) **Diagram-Aware RAG (DA-RAG) Agent:** In RF/Analog design, diagrams commonly encapsulate substantial information—such as schematic topologies, performance curves, and intricate design annotations—that cannot be readily conveyed through textual means. To effectively extract such information, we propose a Diagram-Aware RAG (DA-RAG) wherein real-world circuit textbooks [18], often filled with specialized domain expertise, are methodically converted into Markdown format [19], as illustrated in Figure 5. Unlike conventional RAG-based frameworks that primarily process text, DA-RAG leverages a LLM to transform these diagrams into textual representations, thereby enabling downstream retrieval and capture design requirements comprehensively.

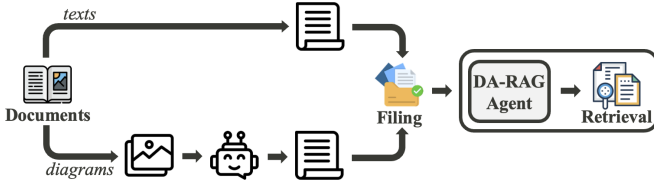


Fig. 5: Overview of Diagram-Aware RAG (DA-RAG).

3) **Executor Agent:** Similar to a compiler, Executor Agent functions as a self-referential system that syntactically validates generated code and circuit descriptions, ensuring their correctness and executability before passing them to subsequent stages of the workflow.

IV. EXPERIMENT RESULTS

To evaluate the effectiveness of our framework, we compared MenTeR against several baseline approaches, including individual LLMs (single-agent) and multi-agent systems. We evaluated them on 24 standard analog design tasks from [3], ranging from elementary designs to complex multi-block design challenges. Furthermore, we deployed our workflow to solve a CMOS Bandgap Reference circuit, a practical task encountered in real-world implementations, to validate MenTeR’s performance in industrial applications.

A. Experimental Setup

We evaluated all methods by generating circuit netlists and checking whether the produced solution passed functional validation. Specifically, we considered the $pass@k$ metric,

which measures the probability of obtaining at least one correct solution within k attempts:

$$pass@k = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \quad (1)$$

where:

- n is the total number of solutions generated,
- c is the number of correct solutions,
- k is the success threshold (number of attempts).

A circuit is deemed “correct” if it meets the specified performance criteria (e.g., gain, power consumption, linearity, bandwidth, etc.) To ensure consistency, all solutions generated were further inspected with basic electrical rule checks and circuit simulations, verifying that the netlists could be functionally simulated without errors.

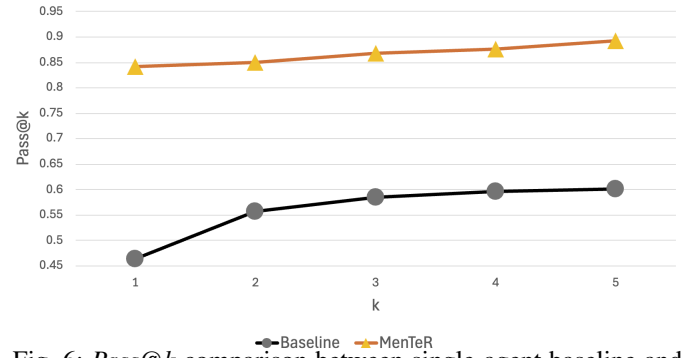


Fig. 6: $Pass@k$ comparison between single-agent baseline and MenTeR.

B. Comparison Results

Table I presents our experimental results, with testing limited to five attempts per task for cost effectiveness. We used GPT-4o as the backbone model for all our approaches to maintain cost efficiency and ensure fair comparison with single-agent methods. For baseline single-agent evaluations, we selected AnalogCoder [3] implemented with distilled model variants (e.g. OpenAI o3-mini [23], DeepSeek-R1 distilled 32B [24]) rather than state-of-the-art reasoning models (e.g. OpenAI o1 [21], DeepSeek-R1 671B [22]), considering the substantial inference costs associated with the latter.

1) **Overall Performance:** MenTeR demonstrates strong performance in various analog design tasks, achieving an average $Pass@1$ rate of 84.2%. Specifically, MenTeR consistently solves basic circuits (e.g., Task 1–6) within a single attempt and outperforms other methods in many advanced tasks requiring multi-block circuit interplay. This significantly outperforms the single-agent baseline, even when the baseline is given multiple attempts (at least five tries per task), as illustrated in Figure 6.

Remarkably, MenTeR achieves a successful $Pass@1$ rate of at least 80% and a perfect 100% at $Pass@5$ rate for complex tasks (e.g., Task 16–24) such as Schmitt trigger, Voltage-Controlled Oscillator, and low-voltage BGR circuit. This indicates its potential to assist real-world complex analog design tasks.

Table I: Performance comparison (in percentage) on 24 analog design tasks and a BGR problem. The highest scores for the hard tasks and average performance are highlighted as **bold**. (Task 1–8: Easy, Task 9–13: Medium, Task 14–24 & BGR: Hard)

Method	AnalogCoder (Single-Agent)						Multi-Agent Workflow (ours)		MenTeR w/o DA-RAG (ours)		MenTeR w/o CoS (ours)		MenTeR (ours)	
Model	DeepSeek R1 Distill 32B		GPT-4o		GPT-o3-mini		GPT-4o							
Task	Pass@1	Pass@5	Pass@1	Pass@5	Pass@1	Pass@5	Pass@1	Pass@5	Pass@1	Pass@5	Pass@1	Pass@5	Pass@1	Pass@5
1	100.0	100.0	100.0	100.0	60.0	80.0	100.0	100.0	80.0	100.0	100.0	100.0	100.0	100.0
2	60.0	100.0	100.0	100.0	40.0	100.0	100.0	100.0	100.0	100.0	80.0	100.0	100.0	100.0
3	20.0	60.0	100.0	100.0	80.0	100.0	80.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
4	40.0	40.0	40.0	80.0	60.0	100.0	80.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
5	20.0	40.0	0.0	40.0	80.0	80.0	100.0	100.0	80.0	100.0	80.0	100.0	100.0	100.0
6	100.0	100.0	100.0	100.0	60.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
7	80.0	80.0	80.0	100.0	20.0	100.0	100.0	100.0	80.0	100.0	100.0	100.0	80.0	80.0
8	60.0	60.0	100.0	100.0	60.0	80.0	100.0	100.0	100.0	100.0	80.0	80.0	100.0	100.0
9	20.0	60.0	60.0	100.0	60.0	100.0	100.0	100.0	100.0	100.0	40.0	60.0	100.0	100.0
10	0.0	60.0	60.0	100.0	80.0	100.0	100.0	100.0	60.0	60.0	100.0	100.0	100.0	100.0
11	0.0	0.0	0.0	20.0	20.0	100.0	20.0	40.0	20.0	60.0	20.0	20.0	40.0	60.0
12	0.0	0.0	0.0	0.0	20.0	80.0	0.0	0.0	40.0	60.0	20.0	60.0	40.0	40.0
13	0.0	0.0	40.0	60.0	60.0	100.0	20.0	20.0	0.0	0.0	20.0	40.0	20.0	20.0
14	0.0	0.0	0.0	60.0	0.0	60.0	20.0	20.0	40.0	40.0	20.0	40.0	40.0	60.0
15	0.0	0.0	0.0	0.0	20.0	20.0	20.0	20.0	0.0	0.0	20.0	40.0	20.0	80.0
16	60.0	60.0	80.0	80.0	40.0	80.0	100.0	100.0	60.0	60.0	100.0	100.0	100.0	100.0
17	60.0	60.0	20.0	20.0	0.0	40.0	60.0	60.0	100.0	100.0	100.0	100.0	100.0	100.0
18	20.0	20.0	20.0	20.0	60.0	100.0	80.0	80.0	100.0	100.0	100.0	100.0	100.0	100.0
19	60.0	60.0	60.0	60.0	60.0	100.0	60.0	60.0	100.0	100.0	100.0	100.0	100.0	100.0
20	60.0	60.0	20.0	20.0	20.0	60.0	100.0	100.0	100.0	100.0	80.0	80.0	100.0	100.0
21	40.0	40.0	60.0	60.0	40.0	80.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
22	40.0	40.0	40.0	40.0	40.0	80.0	60.0	60.0	80.0	80.0	100.0	100.0	100.0	100.0
23	0.0	0.0	5.0	60.0	60.0	80.0	100.0	100.0	100.0	100.0	20.0	60.0	100.0	100.0
24	0.0	0.0	10.0	20.0	20.0	40.0	0.0	20.0	60.0	60.0	0.0	0.0	80.0	100.0
Avg	35.5	43.3	45.6	60.0	44.2	81.7	70.8	74.2	75.0	80.0	70.0	78.3	84.2	89.2
BGR	0.0	0.0	0.0	0.0	0.0	0.0	60.0	100.0	60.0	80.0	60.0	80.0	80.0	100.0

C. Ablation Studies

To isolate the contribution of each MenTeR component, we conducted our ablation study to examine the contributions of MenTeR’s key components, as demonstrated in Table II.

- **Without DA-RAG:** Replacing the DA-RAG agent inside *PI Agent* and *Testbench Agent* with a generic approach decreased the average pass@1 by about 9%. This performance degradation highlighted LLM knowledge limitations when operating without specialized domain know-how. Without access to task-relevant documents, it failed to effectively address advanced analog design tasks that require specific circuit knowledge beyond the model’s inherent capabilities.
- **Without CoS Reasoning:** Removing CoS logic left the multi-agent system with a more “flat” approach to design. Pass@1 dropped significantly (maximum by 14%), especially on tasks with multiple nested constraints. Notably, we observed that the Pass@1 rate of MenTeR without CoS fell below the multi-agent baseline, suggesting that without structured reasoning guidance, the system becomes overwhelmed by retrieved document information and fails to prioritize relevant design knowledge.

Table II: Ablation Studies Results for MenTeR Components.

Method	Avg. Pass@1 (%)	Avg. Pass@5 (%)
Multi-Agent Workflow	70.8	74.2
MenTeR w/o DA-RAG	75.0	80.0
MenTeR w/o CoS	70.0	78.3
Full MenTeR	84.2	89.2

These findings substantiate the necessity of both DA-RAG

and CoS reasoning within the MenTeR framework, demonstrating how each component addresses specific limitations in LLM-based analog circuit design.

V. CONCLUSION

In this work, we propose MenTeR, a fully-automated multi-agent workflow for RF/Analog circuit design that scales effectively to system-level circuit design. By leveraging techniques including Diagram-Aware Retrieval-Augmented Generation (DA-RAG), Chain-of-Stage (CoS) reasoning, and self-referential mechanisms, MenTeR successfully addresses various fundamental circuit blocks without requiring human intervention or extensive fine-tuning. Specifically, MenTeR achieves significantly better performance when handling increasingly complex circuits compared to single-agent approaches, validating its potential for scaling up to system-level circuits and integration with contemporary design flows.

Opportunities and Future work. We acknowledge that MenTeR’s capabilities remain highly correlated with the underlying large language models that power its agents. For more complex analog design tasks, domain-adapted reasoning models specifically fine-tuned for analog design could be a promising direction for future development. Throughout MenTeR’s operation, we have deliberately structured the system to collect high-quality design data, with the goal of building a comprehensive analog design reasoning dataset. We expect this framework will facilitate the development of LLM-driven EDA tools for analog circuits, paving the way towards collaborative teamwork between human analog designers and an RF/Analog copilot.

REFERENCES

- [1] M. Liu, T. Ene, R. Kirby, C. Cheng, N. Pinckney, R. Liang, J. Alben, H. Anand, S. Banerjee, I. Bayraktaroglu, B. Bhaskaran, B. Catanzaro, A. Chaudhuri, S. Clay, B. Dally, L. Dang, P. Deshpande, S. Dhodhi, S. Halepete, E. Hill, J. Hu, S. Jain, B. Khailany, K. Kunal, X. Li, H. Liu, S. Oberman, S. Omar, S. Pratty, A. Sarkar, Z. Shao, H. Sun, P. P. Suthar, V. Tej, K. Xu, and H. Ren, "Chipnemo: Domain-adapted llms for chip design," 2023.
- [2] J. Blocklove, S. Garg, R. Karri, and H. Pearce, "Chip-chat: Challenges and opportunities in conversational hardware design," arXiv preprint arXiv:2305.13243, 2023.
- [3] 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.
- [4] 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.
- [5] Y. Shi, Z. Tao, Y. Gao, T. Zhou, C. Chang, T. Wang, B. Chen, G. Zhang, A. Liu, Z. Yu, T. Lin, L. He, "AMSnet-KG: A Netlist Dataset for LLM-based AMS Circuit Auto-Design Using Knowledge Graph RAG", arXiv preprint arXiv:2411.13560, 2024.
- [6] Y. Deng, W. Zhang, Z. Chen, and Q. Gu, "Rephrase and respond: Let large language models ask better questions for themselves" arXiv preprint arXiv:2311.04205
- [7] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou, "Chain of thought prompting elicits reasoning in large language models," arXiv preprint arXiv:2201.11903
- [8] 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," CoRR, vol. abs/2305.10601, 2023.
- [9] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Kuttler, M. Lewis, W.-t. Yih, T. Rocktaschel et al., "Retrieval augmented generation for knowledge-intensive nlp tasks," Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474, 2020.
- [10] B. Liu et al., "LayoutCopilot: An LLM-Powered Multi-Agent Collaborative Framework for Interactive Analog Layout Design," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, doi: 10.1109/TCAD.2025.3529805.
- [11] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe, Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, Advances in Neural Information Processing Systems, volume 35, pages 27730–27744. Curran Associates, Inc., 2022.
- [12] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, Direct preference optimization: Your language model is secretly a reward model. In NeurIPS, 2023.
- [13] Y. Meng, M. Xia, and D. Chen, SimPO: Simple Preference Optimization with a Reference-Free Reward. In NeurIPS, 2024.
- [14] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948.
- [15] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, et al. 2024. Autogen: Enabling next-gen LLM applications via multi-agent conversation. In ICLR 2024 Workshop on Large Language Model (LLM) Agents.
- [16] L. Song, J. Liu, J. Zhang, S. Zhang, A. Luo, S. Wang, Q. Wu, and C. Wang, Adaptive in-conversation team building for language model agents. arXiv preprint arXiv:2405.19425, 2024.
- [17] B. Razavi. 2017. Design of Analog CMOS Integrated Circuits (2nd. ed.). McGraw-Hill, Inc., USA.
- [18] B. Razavi, "The Design of a Low-Voltage Bandgap Reference [The Analog Mind]," in IEEE Solid-State Circuits Magazine, vol. 13, no. 3, pp. 6–16, Summer 2021, doi: 10.1109/MSSC.2021.3088963.
- [19] Vik Paruchuri, Marker, 2023, GitHub, <https://github.com/VikParuchuri/marker>
- [20] I. M. Filanovsky and H. Baltes, "CMOS Schmitt trigger design," in IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 41, no. 1, pp. 46–49, Jan. 1994, doi: 10.1109/81.260219.
- [21] OpenAI o1, <https://openai.com/o1/>
- [22] DeepSeek R1, <https://huggingface.co/deepseek-ai/DeepSeek-R1>
- [23] OpenAI o3-mini, <https://openai.com/index/openai-o3-mini/>
- [24] DeepSeek R1 Distilled Qwen 32B, <https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-32B/>
- [25] AICB: Analog Integrated Circuit Benchmark, <https://github.com/treeleaves30760/AICB-Analog-Integrated-Circuit-Benchmark/>
- [26] C. Packer, S. Wooder, K. Lin, et al. MemGPT: Towards LLMs as Operating Systems. arXiv preprint arXiv:2310.08560, 2023.

ACKNOWLEDGEMENT

We would like to thank Wei-Chen Chien and Min-Chun Wu for their valuable consultation on the agent framework.

APPENDIX A MENTER STABILITY

We evaluated MenTeR using different backbone models, as summarized in Table III. MenTeR achieves at least a 77% pass@1 rate and 85% pass@5 rate across both models, demonstrating its stable performance. Notably, GPT-4o outperforms GPT-4.1, with higher average pass rates in both metrics. The slightly lower performance of GPT-4.1 may be attributed to its relative deficiency in Analog IC knowledge, which we discuss further in Appendix E.

Table III: MenTeR with different backbone models.

MenTeR Model	Avg. Pass@1 (%)	Avg. Pass@5 (%)
GPT-4o	84.2	89.2
GPT-4.1	77.5	85.8

APPENDIX B COMPARISON WITH REAL-WORLD DESIGN

For the Phase-Locked Loop (PLL) task, we compared AI-generated results with a human-designed reference. Our analysis revealed discrepancies between circuits that passed the benchmark validation and those that were functionally correct.

For instance, Figure 7 shows a PLL implementation that was deemed correct by the benchmark criteria but contained errors upon schematic inspection. These errors primarily fell into two categories: incorrectly defined subcircuits (Frequency Detector (FD), Voltage Controlled Oscillator (VCO)), and missing wire connections (VCO to FD, FD to Phase Frequency Detector (PFD), and PFD to the reference frequency source). These issues stem from the inherent complexity of the PLL as a hierarchical system-level design. Notably, while individual subcircuits successfully passed DC sweep simulations in isolation, the integrated system as a whole failed to function correctly. This highlights the challenges of fully automating the generation of complex system-level designs using LLM-based RF/Analog design tools.

To further deploy MenTeR into real-world design environments, maintaining a "human-in-the-loop" approach remains essential. Such human oversight is particularly critical for complex system-level designs like PLL, where subtle integration errors may escape automated validation. Nevertheless, as the analog design reasoning capabilities of LLMs continue to advance, we can reasonably expect to reduce human intervention significantly, allowing designers to focus their expertise on system-level verification and optimization rather than routine circuit implementation tasks.

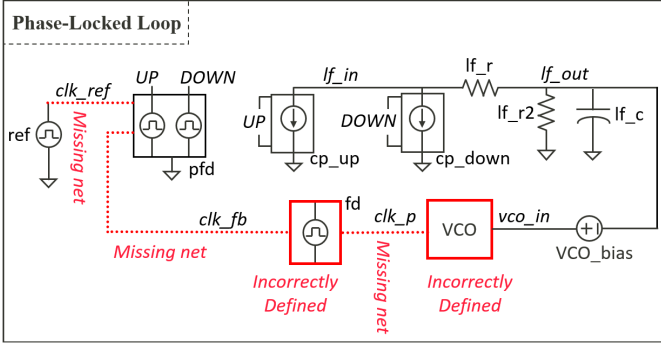


Fig. 7: Schematic of an AI generated Phase-Locked Loop (PLL).

APPENDIX C ENGINEERING CHALLENGES

In our experiment, we encountered several engineering challenges that impacted efficiency and resource utilization. In this section, we share these challenges and their corresponding solutions to assist future LLM-based analog design research.

Token Limits. The primary issue we encountered when testing on [3] was exceeding the token limits, which led to frequent crashes and interfered with our experimental progress. Upon investigation, we identified that the root cause stemmed from inefficient chat history parsing techniques that included the whole conversation texts in each iteration, which occupied the context window rapidly. To resolve this limitation, we modified our approach to truncate the chat history in each agent, retaining only the initial prompt and the last message. This optimization effectively reduced token consumption and eliminated system crashes.

In addition, we acknowledge the concerns raised by reviewers regarding potential information loss when adopting context truncation techniques. To address this issue, we believe that approaches similar to the memory management system proposed by [26] represent promising avenues to solve the "lost-in-the-middle" issue.

LLM Instability. Another substantial obstacle encountered was the instability introduced by iterative updates to the underlying large language model. Although no official statement has confirmed this phenomenon, there exists a widely discussed concern within the research community regarding sudden shifts in model behavior and performance after platform updates. Our own observations were consistent with these anecdotal reports; at various points in the experimental process, the model's generative consistency and accuracy appeared to deteriorate following version changes and seed randomness. While further investigation is required to systematically verify the extent of this issue, the unanticipated variability in model output poses significant challenges for maintaining reproducibility and fine-tuning results in LLM-based analog design research.

Execution Errors. We experienced a notably high rate of execution errors when reproducing single-agent results. This issue significantly interrupted the workflow as the generated code was often "unrunnable," preventing it from reaching the

testbench validation stage. To mitigate this issue, we incorporated a self-referential technique into both multi-agent and MenTeR, ensuring all code is executable before submission. This technique substantially decreased execution error rates, which led to the stability of our methodologies (as illustrated in Figure 6), improving deployment efficiency of the overall system.

Repetitive Error Loops. We observed that certain errors appeared repeatedly during our experimental iterations; LLMs continuously encountering the same issues and becoming stuck, even after receiving specific instructions to avoid them. This cycle led to task failures and resulted in substantial resource waste. We determined this occurred due to inherent limitations of LLMs. Consequently, we implemented an early-detection mechanism to identify such recurring errors, automatically terminating the chat when these patterns are detected. In future work, "preventing LLMs from repeating the same mistakes" represents an important research direction that needs further exploration.

Token Efficiency. Lastly, we compare the token efficiency of the single-agent approach and MenTeR, both utilizing GPT-4o as the backbone model. In this context, prompt tokens refer to the input tokens provided to the model, while completion tokens denote the output tokens generated by the model. As shown in Table IV, MenTeR consumes more tokens for the relatively simple Task 1. However, for the more challenging Task 24, the single-agent method exhibits a substantial increase in token usage, yet still fails to solve the task.

This highlights that while the single-agent approach remains effective for straightforward tasks, its capability diminishes as task complexity increases. Balancing token efficiency and problem-solving capability, particularly in RF/Analog design, remains an important direction for future research.

Table IV: Token Comparison between Single-Agent and MenTeR. (* means fail)

Token	Single-Agent		MenTeR	
	Prompt	Completion	Prompt	Completion
Task 1 (Easy)	1330	269	53991	2938
Task 24 (Hard)	98173	43310*	39038	2428

APPENDIX D ALTERNATIVE SCHEMATIC EXPLORATION

During our experiments, we noticed that the same problem specification can yield multiple valid solutions with slightly different transistor arrangements or bias schemes. For instance, in the tasks labeled as "Problem 20," the system generated two distinct netlists (20-0 vs. 20-1), all claiming to meet similar specifications.

We hypothesize that this behavior arises because the given problem constraints are not sufficiently strict to enforce a single unique topology or bias point. Despite leading to multiple functional solutions, such variability also highlights the LLM's capacity to explore alternative schematics. Below, we include two representative netlists, which aim to implement a simple op-amp-based adder.

A. Problem 20: Op-Amp Adder

Netlist #20-0

```

1 .title Opamp Adder
2 .subckt SingleStageOpamp Vinp Vinn Vout
3 Vdd Vdd 0 5.0
4 Vbias Vbias 0 1.5
5 M1 Voutp Vinp Source3 Source3 nmos_model l=1e-06
  w=5e-05
6 M2 Vout Vinn Source3 Source3 nmos_model l=1e-06 w
  =5e-05
7 M3 Source3 Vbias 0 0 nmos_model l=1e-06 w=0.0001
8 M4 Voutp Voutp Vdd Vdd pmos_model l=1e-06 w
  =0.0001
9 M5 Vout Voutp Vdd Vdd pmos_model l=1e-06 w=0.0001
10 .model nmos_model nmos (kp=0.0001 level=1 vto
  =0.5)
11 .model pmos_model pmos (kp=5e-05 level=1 vto
  =-0.5)
12 .ends SingleStageOpamp
13
14 Vbias V_bias 0 1.79V
15 Vref Vref 0 1.79V
16 Vin1 Vin1 0 1.79V
17 Vin2 Vin2 0 1.915V
18 X1 V_bias Vinn Vout SingleStageOpamp
19 R1 Vin1 Vinn 20kOhm
20 R2 Vin2 Vinn 20kOhm
21 Rf Vout Vinn 20kOhm
22 Rref Vref Vinn 20kOhm

```

Netlist #20-1

```

1 .title Opamp Adder: Vout = -(Vin1+Vin2)
2 .subckt SingleStageOpamp Vinp Vinn Vout
3 Vdd Vdd 0 5.0
4 Vbias Vbias 0 1.5
5 M1 Voutp Vinp Source3 Source3 nmos_model l=1e-06
  w=5e-05
6 M2 Vout Vinn Source3 Source3 nmos_model l=1e-06 w
  =5e-05
7 M3 Source3 Vbias 0 0 nmos_model l=1e-06 w=0.0001
8 M4 Voutp Voutp Vdd Vdd pmos_model l=1e-06 w
  =0.0001
9 M5 Vout Voutp Vdd Vdd pmos_model l=1e-06 w=0.0001
10 .model nmos_model nmos (kp=0.0001 level=1 vto
  =0.5)
11 .model pmos_model pmos (kp=5e-05 level=1 vto
  =-0.5)
12 .ends SingleStageOpamp
13
14 X1 bias inv Vout SingleStageOpamp
15 Vin1 Vin1 0 0V
16 Vin2 Vin2 0 0V
17 Vbias_source bias 0 1.79V
18 R1 Vin1 inv 10kOhm
19 R2 Vin2 inv 10kOhm
20 Rf Vout inv 10kOhm
21 Voffset_source voffset 0 3.58V
22 Roffset voffset inv 3.3333333333333335kOhm

```

Although each netlist relies on similar single-stage operational amplifier blocks, the resistor arrangements differ. These differences stem from multiple ways to achieve the same functional requirement (i.e., summing two input signals).

Overall, our findings indicate that when problem constraints are loosely defined, LLMs can explore multiple design topologies. While each solution may meet the functional specs at a high level, additional constraints (e.g. output swing requirements, matching conditions, noise margins) would be needed to converge on more uniform design outcomes.

APPENDIX E ANALOG IC KNOWLEDGE

In order to investigate the root causes of differing performance among LLM-based analog design approaches, we developed the Analog IC Benchmark (AICB) [25]. This dataset consists of **300 multiple-choice questions** derived from a standard analog IC (AIC) textbook, with each question providing **four possible answers**.

A. Dataset Introduction

To illustrate the format of AICB, we can consider the following example question:

Question: What is the primary function of a sample-and-hold circuit in an ADC?

- (A) To amplify the input signal
- (B) To convert the analog signal to digital
- (C) To hold the input signal constant in conversion
- (D) To filter the input signal

The correct answer is **C**, reflecting a key concept in data conversion systems.

B. Performance Results

Table V summarizes the accuracies achieved by these models on the AICB dataset. As indicated, GPT-o3-mini attained the highest accuracy, closely followed by GPT-4o, while DeepSeek R1 Distill 32B showed noticeably lower performance.

Table V: Performance of LLMs on the AICB Dataset

Model	Accuracy (%)
DeepSeek R1 Distill 32B	75.7
GPT-4o	85.0
GPT-o3-mini	91.3
GPT-4.1	81.1

C. Key Observations

Instruction-Following Challenges in DeepSeek R1 Distill 32B. We identify that DeepSeek R1 Distill 32B generates inconsistent output formatting (e.g., generating ****Answer****: A or **### Answer A** instead of the prescribed tag format), which contributes to the poor performance of the model within an agent-based system, as such systems often rely on strict text formats for downstream task coordination. Consequently, DeepSeek R1 Distill 32B’s difficulties in adhering to instructions partially explain its poor results in Table I when paired with the benchmark [3].

D. Limitations and Future Directions

While AICB provides a preliminary means of assessing LLM proficiency in analog IC domains, it does not fully capture the complexities encountered in real-world circuit design. As a result, we plan to expand AICB by incorporating more challenging and open-ended questions, aligned with the multifaceted nature of industrial analog design tasks. Such an enhanced benchmark would offer deeper insights into each model’s capabilities and limitations, facilitating more robust research and development of LLM-based analog design tools.