The Knowledge Engineering Review

RESEARCH ARTICLE

Open Access

https://doi.org/10.48130/ker-0026-0012

The Knowledge Engineering Review **2026**, 41: e011

Structure–performance coupled analog circuit reasoning with Large Language Models

Jiaqing Zhou, Hao Li*, Liqing Zhou and Hao Jiang

Electronic Information School, Wuhan University, Wuhan, Hubei 430072, China

* Correspondence: whulh@whu.edu.cn (Li H)

Abstract

Analog circuits are essential for analog signal processing, yet their design requires structure–performance reasoning over the coupled relationship between topology, device, and performance. Recent advances in Large Language Models (LLMs) have inspired their use in analog circuit design, where SPICE netlists map to text-generation tasks. However, current LLMs fail to design like human designers. They exhibit inconsistent hierarchical design, succeeding in circuit devices but failing to design them in derived circuits, and show limited circuit modification, struggling to make minor modifications to classic circuits. Considering that LLMs excel at well-defined, low-complexity tasks, we improve the performance by decomposing the circuit design process and focusing on structure–performance coupled reasoning. We further propose ACR-Agent, a multi-agent framework that mirrors expert workflows by decomposing circuit reasoning into choosing, pre-analysis, and analysis steps. We validate its effectiveness by using ACR-Bench, a benchmark that covers a full syllabus of circuit topics with circuit analysis tasks in four difficulties. Experiments on ACR-Bench show that ACR-Agent significantly improves reasoning accuracy and consistency compared to baselines. Downstream analog circuit design validation shows ACR-Agent's integration enhances existing circuit design LLMs. Together, ACR-Agent and ACR-Bench provide a foundation for enhancing LLMs in canonical analog circuit reasoning, encouraging transfer potential to circuit design.

Keywords: Analog circuit, Large Language Models, Multi-agent, Structure–performance coupled reasoning, Circuit reasoning benchmark

Citation: Zhou J, Li H, Zhou L, Jiang H. 2026. Structure–performance coupled analog circuit reasoning with Large Language Models. *The Knowledge Engineering Review* 41: e011 https://doi.org/10.48130/ker-0026-0012

1 Introduction

Analog circuits anchor integrated systems by precisely processing continuous-valued signals (analog signals)[1–3]. They perform designed conversions on analog signals, such as amplification, filtering, and operation, and thereby establish the gain, bandwidth, and binarization required for downstream processing. Such conversion underpins reliable data acquisition and signal processing in domains including wireless communication, autonomous vehicle perception, energy-constrained biomedical implants, and high-resolution imaging[3,4].

As large language models continue to achieve impressive results on many challenging tasks, researchers have begun to explore their potential in analog circuit design[5,6]. In this domain, engineers describe circuits in SPICE (Simulation Program with Integrated Circuit Emphasis)[7], a plain-text netlist that specifies each device, its connections, and key parameters. With this structured representation, Large Language Models (LLMs) can directly generate complete designs, defining both the topology and the devices in text[8,9]. Just as the successful application of LLMs to code generation has lowered the entry barrier and decreased the workload by more than 50% for software development[10], their application to analog circuit design is expected to further reduce time and labor costs. With exceptional semantic comprehension capabilities and PhD-level expertise, LLMs have great potential for application in the specification and schematic design within the analog circuit design flow shown in Fig. 1.

However, large language models cannot yet work like real analog circuit designers. Their challenges appear most clearly in two aspects: **Inconsistent hierarchical design.** The model can design integrated circuit devices, but it fails to correctly design the same devices in their derived circuits. As shown in Fig. 2a, it can design a single-stage differential common-source operational amplifier (op-amp), yet it cannot be implemented in an adder. **Limited circuit modification.** When faced with requirements for slightly adapted classic circuits, the model performs poorly and fails to modify the circuit. As shown in Fig. 2b, it cannot replace the ideal current source with a PMOS current source according to the requirement.

Given that current LLMs exhibit satisfactory performance when handling low-complexity tasks with explicit instructions[11,12], we try to improve the performance by decomposing complex design tasks. We divide the design process into text-based analysis from requirements to circuits, and circuit generation in specific formats like SPICE, with our focus on the former, which constitutes the foundation of

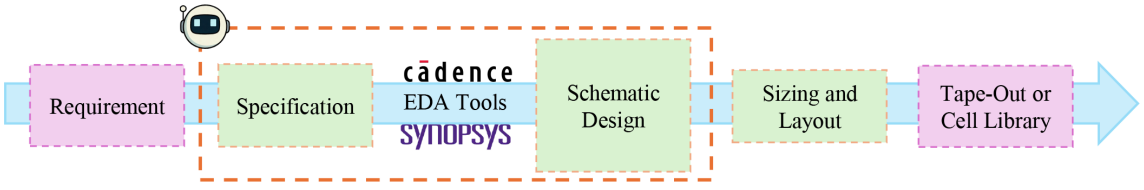

Fig. 1 Overview of the key manual steps in analog circuit design flow. The larger the area of a step block, the higher its difficulty and importance.

https://doi.org/10.48130/ker-0026-0012

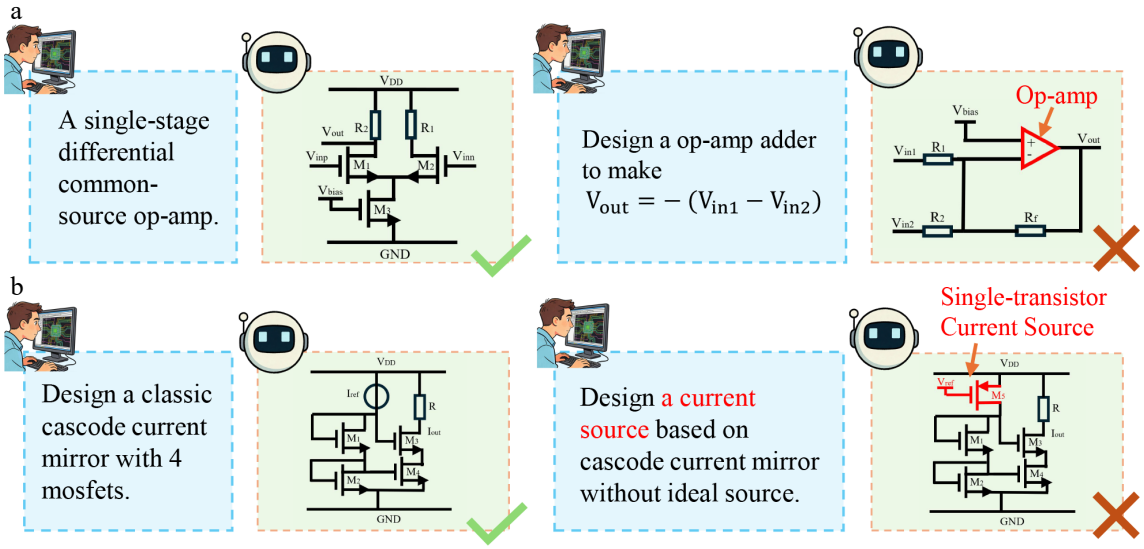

Fig. 2 Typical failures of LLMs in analog circuit design. Red marks core failure causes.

successful design but gains little attention in previous works[8,9,13–15]. Its core is *structure–performance coupled* reasoning: The structure and performance in analog circuits are tightly interdependent; structure determines performance, and likewise, performance requirements determine the designed circuit structure. We further subdivide and enhance this reasoning to improve the overall performance of LLMs in analog circuit design.

In this way, we propose ACR-Agent (Analog Circuit Reasoning Multi-Agent Framework). This framework mirrors the workflow of experienced human designers, who break down circuit analysis into well-defined steps to achieve self-consistent structure–performance solutions. The **Choosing Agent** receives the problem statement, identifies the relevant subcircuits, and retrieves useful information from a domain-specific knowledge base. The **Pre-analysis Agent** examines the selected netlist, determines intermediate targets that must be derived before reaching the final result, and splits circuit device groups by their functions. Finally, the **Analysis Agent** performs the detailed derivation, reasoning over circuit topology and device behavior, and outputs a final solution. By decomposing the overall reasoning process into smaller, specialized steps, ACR-Agent achieves a deeper and more structured understanding of the structure–performance relationship in analog circuits, leading to more accurate and interpretable analysis.

To validate the effectiveness of ACR-Agent, we introduce the ACR-Bench (Analog Circuit Reasoning Benchmark), a benchmark dedicated to evaluating analog circuit analysis with the structure–performance relationship. It is constructed from authoritative textbooks in analog and integrated circuit design and spans the essential syllabus with circuit analysis tasks in 10 topics[3]. Each task provides a textual description, a SPICE netlist, and a LaTeX expression solution. The tasks are organized into four levels reflecting the growing analysis complexity in larger topologies. ACR-Bench thus offers a systematic and reproducible way to assess the capacity for textbook-level structure–performance coupled reasoning, which is the foundation of real analog circuit design.

This paper makes two main contributions:

1. We propose **ACR-Agent**, a multi-agent LLM framework that mirrors expert workflows by dividing the reasoning process into choosing, pre-analysis, and analysis steps, and it improves the structure–performance coupled analog circuit reasoning of LLMs on canonical circuit reasoning tasks.

2. We construct **ACR-Bench** to validate the effectiveness of ACR-Agent, and experiments show that the reasoning accuracy and consistency are improved by 19% and 24% compared with baselines. We carry out **downstream analog circuit design validation** and prove ACR-Agent's improvement to existing analog circuit design LLMs and show promising transfer potential toward practical usage.

2 Related works

2.1 Artificial intelligence in analog circuits

Recent advances in artificial intelligence have brought analog circuit design into focus for machine learning researchers[16,17]. Early studies framed circuit synthesis as the generation of graphs, where graph neural networks predict both devices and nets[18,19]. The arrival of large language models made it possible to encode circuits as token sequences so that autoregressive or fine-tuned models could output topologies[20–22]. These pipelines still require highly structured and domain-specific prompts and provide limited interactivity[22,23]. In response, recent work employs general-purpose language models that embed SPICE netlists into prompts and produce circuits in a style similar to code generation, while using retrieval-augmented knowledge and the composition of subcircuits to improve results[7–9,13,24,25]. The CIRCUIT benchmark gives a snapshot of headline improvements[26], yet systematic studies of how these models reason about device relationships, and how that reasoning can be enhanced, are still missing.

2.2 Large Language Model reasoning

A Large Language Model (LLM) is a deep neural network with massive parameters and excels at understanding, generating, and reasoning about natural language[27,28]. As the reasoning capabilities and methods of LLMs continue to advance[29,30], they demonstrate robust performance on highly challenging mathematical and real-world reasoning tasks[31,32]. For a subset of professional tasks, the limitations in LLMs' reasoning abilities within specific domains constrain their task performance[33,34]. Circuit analysis and generation are one of these complex reasoning and generative tasks for LLMs[6], making the study and improvement of their analog circuit reasoning performance and capacity highly significant.

3 ACR-Agent

ACR-Agent is a multi-agent framework designed to simulate the reasoning process of human experts and improve LLMs' ability to perform structure–performance coupled reasoning in analog circuit tasks. It consists of three LLM-based agents with chain-of-thought prompting[29]: the Choosing Agent, the Pre-analysis Agent, and the Analysis Agent. They collaborate following the workflow shown in Fig. 3.

3.1 Choosing Agent

SPICE netlists are not traditional natural language, so conventional retrieval-augmented generation (RAG)[35] is ineffective and often introduces noise that harms reasoning[36]. To address this, the Choosing Agent uses two structured knowledge resources: a Concept Library and a Subcircuit Library. Based on the task description, keywords, and selection tips, it retrieves relevant entries for use by the Pre-analysis Agent and Analysis Agent. Details of libraries are shown in Appendix A1.

3.1.1 Choosing workflow

Due to LLMs' inherent limitations, it is challenging to simultaneously perform simple circuit analysis and process a large volume of complex selection tips in a single output. This limitation will become even more pronounced as libraries continue to expand in the future. To address this challenge, we have designed a selection workflow within the Choosing Agent. First, knowledge entries are divided into many units. Entry units are generated automatically and randomly, with the only constraint that each unit contains no more than three knowledge entries. For each unit, the corresponding tasks, keywords, and selection tips are put into a Unit Selection LLM. The Unit Selection LLM analyzes the input and makes a selection in its unit of knowledge entries. Its selection includes the keywords of the chosen entries and a brief reason. Then, the selections made by these Unit Selection LLMs, along with the original tasks, all keywords, and selection tips, are input into a Review and Summary LLM through its context window. If the Review and Summary LLM agree with all selections from the Unit Selection LLMs, its output is the final choice of the Choosing Agent. If there is a disagreement, another Review and Summary LLM engages in a rebuttal process with the initial one until a consensus is reached, and this consensus is the final choice of the Choosing Agent. The review and summary LLMs are prompted to output '**I agree.**' or '**I disagree.**' at the beginning of their output, which shows their attitude and acts as a flag to begin or end the rebuttal process. The Choosing Agent finally outputs a group of keywords of chosen entries, and then the program automatically generates a knowledge string by combining them.

3.1.2 Concept Library

A Concept Library is introduced to address the issue of inconsistent hierarchical understanding of the structure–performance relationship in an analog circuit. By reintroducing knowledge to supplement the LLM's inherent knowledge, it enables LLMs to use the conceptual knowledge from the reasoning prompts, which bridge the cognitive gaps between devices in different circuit hierarchical levels. Each entry in the Concept Library defines a concept and provides an associated analysis method from the textbook[3]. It also comes with keywords and selection tips, which are shown in the Choosing Agent's prompt for reference.

3.1.3 Subcircuit Library

A Subcircuit Library is introduced to address the issue of limited generalization ability. The poor ability of LLMs to modify classic circuits indicates that they may lack information about inner structure–performance relationships of these classic circuits, treat these circuits as a black box, and recite their structures. The clear circuit structures and their detailed analytical results provided in the subcircuit entries enable LLMs to treat these classic circuits as an explainable composition of circuit devices, thereby using them in circuit analysis and making them more flexible. Each subcircuit entry contains a subcircuit in SPICE netlists and its corresponding parametric analysis from the textbook[3], which is an analysis without parameter calculations. It also comes with its keywords and selection tips, which are provided in the Choosing Agent's prompt for reference.

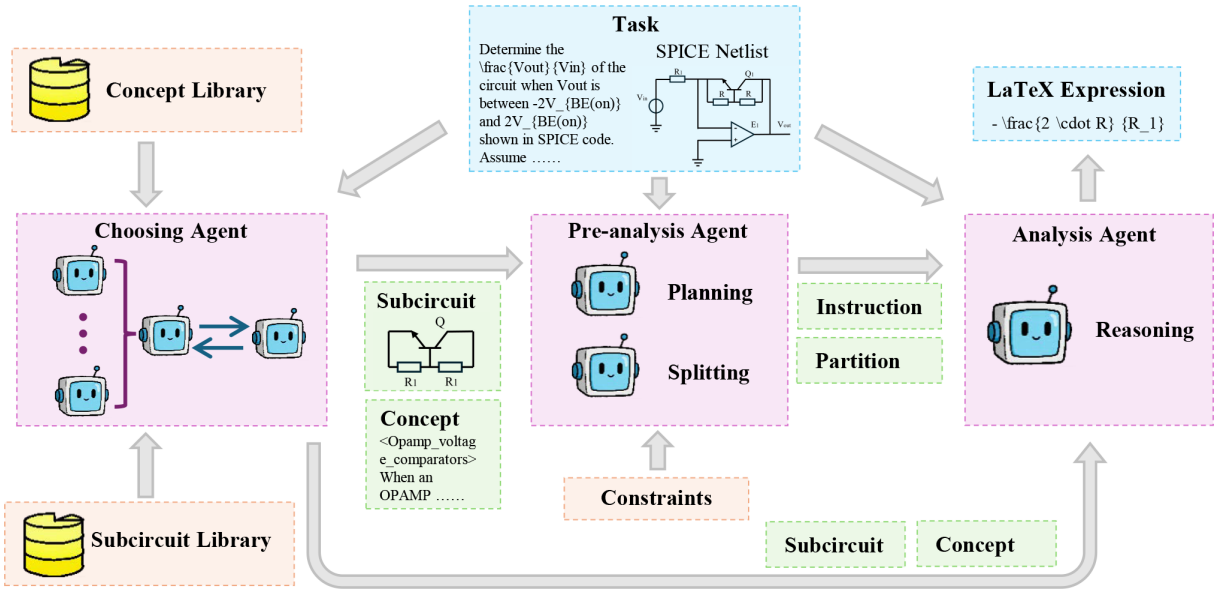

Fig. 3 The overview of ACR-Agent. The circuit analysis task is sequentially processed by the Choosing Agent and Pre-analysis Agent. It is sent to the Analysis Agent along with the selected concept and subcircuit entries, and the generated sequence of sub-tasks and function groups of circuit devices, ultimately generating LaTeX expressions. The circuits in Task and Subcircuit are all SPICE Netlists, which is simplified by circuit diagrams in the figure.

https://doi.org/10.48130/ker-0026-0012

3.2 Pre-analysis Agent

Due to the integration of analog circuits and the complexity of their structure–performance coupled reasoning, direct reasoning imposes high requirements on large language models (LLMs). The reasoning process requires LLMs to accurately identify circuit devices relevant to the problem and quickly find correct reasoning paths. In contrast, human experts typically perform an overall functional analysis of the circuit and plan the analysis scheme based on existing experience before initiating local, detailed analysis. Like human experts, we design a Pre-analysis Agent to accomplish reasoning, planning, and functional device group splitting, and send this information to an Analysis Agent to help its reasoning.

3.2.1 Planning LLM

Analog circuit structure–performance coupled reasoning is a complex reasoning process that requires the combination of general circuit analysis and local reasoning. Planning and separating the problem into simple steps is essential for complex reasoning and multi-agent collaboration[37]. The Planning LLM decomposes a reasoning task into manageable subtasks based on the selected concepts and subcircuits. To avoid overly fine-grained plans that introduce noise, its prompt includes constraints that ensure each subtask remains at an appropriate level of detail. The Planning LLM finally outputs a sequence of subtasks in the form of questions, which ask for key parameters or functions of the circuit. It is prompted to place its sequence of subtasks between the tags `<answer>` and `<answer/>` for extraction, and then concatenate them into the Analysis Agent's prompt between the tags `<BEGIN INSTRUCTION>` and `<END INSTRUCTION>`.

3.2.2 Splitting LLM

Analog circuits are typically composed of multiple functional device groups. For example, an operational amplifier consists of four functional groups: the input stage, intermediate stage, output stage, and biasing circuit[3]. Reasonable separation of these devices enables rapid identification of components relevant to the desired quantity and their corresponding parameters. The Splitting LLM conducts a brief functional analysis of the circuit, partitioning it into distinct functional device groups based on selected concepts and subcircuits. To avoid meaningless over-partitioning, constraints are incorporated into its prompt to regulate the splitting approach. The Splitting LLM finally outputs several functional device groups, along with their main functions and devices. It is prompted to place its functional device groups between the tags `<answer>` and `<answer/>` for extraction, and then concatenate them into the Analysis Agent's prompt between the tags `<BEGIN FUNCTION ANALYSIS>` and `<END FUNCTION ANALYSIS>`.

3.2.3 Constraints

The analog circuit reasoning task is characterized by significant flexibility and inherent complexity. Even a single step of circuit-based reasoning can be decomposed into numerous excessively detailed subtasks, posing challenges to the normal analysis of the Analysis Agent as noise in reasoning. Since some devices collectively fulfill a single function while differing in specific details, an overly granular classification would result in every device being deemed distinct, which is meaningless. Our method incorporates constraints in prompts into the Pre-analysis Agent's LLMs' prompts to ensure they separate functional device groups and generate subtasks at an appropriate level of detail. The constraints are phrased as tips; for example, for numerical symbols already given in the problem, don't generate a question to determine them; assume they are known.

3.3 Analysis Agent

The Analysis Agent is the core reasoning component. It takes the original task, the retrieved concepts and subcircuits, and the planned subtasks, splits functional device groups in its engineered prompt, then performs structured reasoning and generates the final LaTeX-formatted expression solution. The form of the solution can vary with the task, and Fig. 3 presents an example of a circuit analysis task.

3.4 Workflow

As shown in Fig. 3 and Algorithm 1, ACR-Agent's workflow includes three main steps: The Choosing Agent acquires tasks, concept keywords, subcircuit names, and selection tips. Based on these, it selects relevant concepts and subcircuit entries from the Concept Library and Subcircuit Library. The Pre-analysis Agent breaks down tasks and splits circuits under constraints, generating sequences of subtasks and functional device groups according to the task requirements and selected entries. The Analysis Agent reasons and generates the final solutions guided by the subtasks, referencing the selected entries. This design reduces reasoning complexity, improves consistency, and enhances the overall performance of LLMs on analog circuit reasoning tasks.

4 ACR-Bench construction

We have developed ACR-Bench, a benchmark for analog circuit device-relationship reasoning tasks to validate if ACR-Agent is effective, and fill the gap in open-source and dedicated benchmarks in this field. The benchmark construction process is shown in Fig. 4. We select various analog circuit reasoning tasks from authoritative textbooks based on specific criteria. Each chosen analog circuit reasoning task is converted and manually analyzed. Totally, we selected 131 analog circuit reasoning tasks, grouped them up, and constructed our ACR-Bench in the format of question-answer pairs. Our ACR-Bench offers a novel data structure and approach for evaluating LLMs' structure–performance coupled comprehension and reasoning of analog

Algorithm 1. ACR-Agent Workflow.

Require: Analog Circuit Reasoning Task T
Ensure: Final Solution E
Agent 1: Choosing Agent (A_c)
$L \leftarrow L_C \cup L_{SC}$ ▷ Combine Concept Library(L_C) with Subcircuit Library(L_{SC})
$(L_1, L_2, ...L_n) \leftarrow$ Separate(L)
for each L_i in L_n **do**
    $s_i \leftarrow A_{c(i)}(L_i, T)$
end for
$S \leftarrow A_{c(n+1)}(L, s_1, s_2, ..., s_n, T)$
if $S \neq \sum s_i$ **then**
    $S' \leftarrow A_{c(n+2)}(L, s_1, s_2, ..., s_n, T, S)$
    while $S' \neq S$ **do**
        $S \leftarrow$ Debate$(A_{c(n+1)}(L, s_1, ..., s_n, T, S'), A_{c(n+2)}(L, s_1, ..., s_n, T, S))$
    end while
end if
Agent 2: Pre-analysis Agent (A_p)
$L \leftarrow A_{p(split)}(S, T, C)$ ▷ C means Constraints
$P \leftarrow A_{p(plan)}(S, T, C)$
Agent 3: Analysis Agent (A_a)
$E \leftarrow A_a(S, P, L, T)$ **return** E

https://doi.org/10.48130/ker-0026-0012

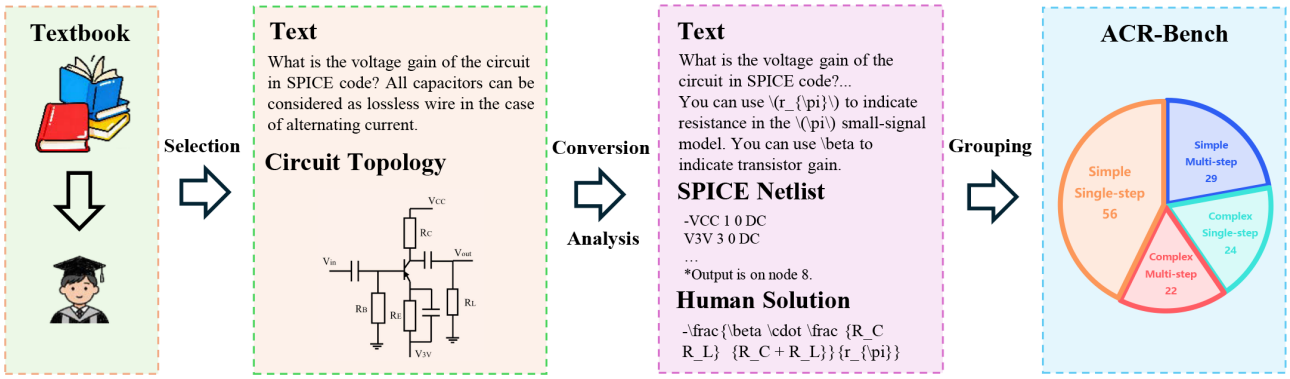

Fig. 4 The construction process of ACR-Bench. The data undergoes selection, conversion, and human analysis, and grouping to form ACR-Bench. The figure omits the depiction of human operators in the conversion, human analysis, and grouping.

Table 1. Comparison between ACR-Bench and other datasets.

Dataset	Analog circuits	SPICE	Data type	Task type	Expression
Masala-chai[38]	✔	✔	Real	Generate	✘
Amsnet[39]	✔	✔	Real	Generate	✘
Amsbench[40]	✔	✘	Real	Reason + design	✘
Eee-bench[41]	✔[1]	✘	Real	Reason	✘
Analogcoder[8]	✔	✔	Real	Design	✘
Analoggenie[22]	✔	✔	Real	Design	✘
Autocircuit-rl[19]	✔	✔	Synthetic	Design[2]	✘
LaMAGIC[20]	✔	✔	Synthetic	Design[2]	✘
Chipnemo[5]	✔[1]	✘	Real	Reason + design	✘
CIRCUIT[26]	✔	✔	Real	Reason	✘
ACR-Bench	✔	✔	**Real**	**Reason**	✔

[1] Mix of analog circuit and other topics. [2] Containing only one type of typical analog circuit. Analog circuits: dataset theme on analog circuits; SPICE: whether analog circuit representations are SPICE netlists; Data type: source type of the dataset; Task type: target task types of the dataset; Expression: whether the output is in expression format. The bold text indicates that ACR-Bench possesses all the characteristics listed in the table.

circuits. The comparison between ACR-Bench and other datasets is presented in Table 1. 'Expression' means the solution is an expression rather than SPICE or a number. 'Generate' in task type means generate SPICE form circuit diagrams. Details of ACR-Bench are shown in Appendix A2.

4.1 Circuit representation

An analog circuit processes continuous-time signals such as sinusoidal voltages to implement specific functions[1,2]. In Fig. 5, a simple amplifier amplifies an input AC signal V_i by a factor of $-\frac{\beta R_C}{R_S + r_\pi}$, where r_π denotes the equivalent resistance of the small-signal model for Q_1, and outputs the result at the V_{out} terminal.

SPICE Netlist: To describe circuit topology in a machine-readable form, we use SPICE, a text-based language widely adopted for analog circuit simulation[7]. In SPICE syntax, each component is defined by its name, connected nodes, and key parameters. In Fig. 5, the amplifier contains five components: V_i, R_S, R_C, V_{CC}, and Q_1. The resistor R_S is written as '|R_S 2 3 500|', where 'R_S' is the component name, '2' and '3' are its connected nodes, and 500Ω is its resistance value. This textual representation preserves the same topology as the original circuit diagram, enabling reasoning tasks to be expressed consistently in ACR-Bench.

4.2 Data source and selection

We selected device–relationship reasoning tasks from widely used textbooks such as Analysis and Design of Analog Integrated Circuits[3], including examples and exercises, and conducted a preliminary screening based on three criteria:

1. Difficulty: select tasks at an undergraduate difficulty level while eliminating those that are too simple or overly challenging
2. Circuit Complexity: select circuits with 4 to 30 components, excluding single-component detailed reasoning tasks
3. Engineering Value: select common and classic circuits as well as their derivatives while excluding those with no practical applicability or representativeness.

In ACR-Bench, we totally select circuit analysis tasks in 10 topics as shown in the Table 2.

4.3 Data processing and grouping

4.3.1 SPICE representation

We converted the circuit topologies into SPICE netlists to embed them into text, which is carried out by an experienced human operator in the Ngspice simulator. Since the benchmark requires analog circuit reasoning tasks, we remove all specific numerical parameters from the

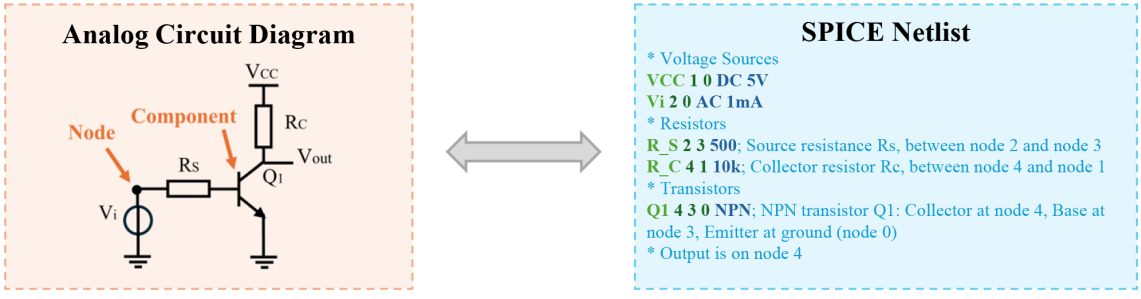

Fig. 5 Representations of the circuit. The example amplifier in diagram can be represented by the SPICE Netlist.

https://doi.org/10.48130/ker-0026-0012

Table 2. Number of ACR-Bench problems in each topic.

Topic	Number
Transistor fundamentals	6
Basic amplifiers	16
Differential amplifiers	10
Active loads	8
Output stages	19
Feedback networks	18
Operational amplifiers	28
Frequency response and compensation	16
Circuit noise	4
Non-linear circuits	6
Total	131

SPICE and questions, retaining only symbolic representations of the values, which alters the syntax of the SPICE netlists. To ensure the integrity of the circuit information in the task, we add comments to the SPICE to further clarify the circuit connections and avoid ambiguity.

4.3.2 Analysis

We manually analyzed selected tasks and provided human solutions. Every task is analyzed by three experienced human operators. When they have different opinions on the solution, we introduce a professional expert to make the final decision. To guarantee that the human solution is the only reasonable solution for the task, we adjusted the parameters that can be used and assumed that all the ungiven parameters and their effects can be neglected in reasoning. This raises the requirements for circuit structure–performance coupled understanding and avoids pre-training data contamination[42]. To ensure final correctness and consistency, the adjusted tasks and solutions have been re-reviewed by two out of the three experienced human operators who previously conducted manual analysis of the original tasks. For example, in the task shown in Fig. 4, we remove all the capacitance parameters and frequencies of the input signal to guarantee the solution $-\frac{\beta \cdot R_C \cdot R_L}{(R_C + R_L) \cdot r_\pi}$ based on the standard small-signal-π model of the transistor is the only reasonable solution, which instructs LLMs to assume that the capacitors in the circuit are ideal and the capacitors in the transistors can be neglected. Such adjustments may simplify the reasoning processes but pose a higher requirement for reasoning flexibility. By enforcing unique symbolic solutions and simplifying secondary effects, ACR-Bench prioritizes standardized reasoning evaluation over real-world design realism. It is designed to assess textbook-level symbolic structure–performance coupled reasoning, the core foundation of real analog circuit design, rather than end-to-end practical analog circuit analysis capability.

4.3.3 Task grouping

We group the tasks into four categories: simple single-step tasks, simple multi-step tasks, complex single-step tasks, and complex multi-step tasks to evaluate reasoning on different complexity of structure–performance coupled reasoning. The criterion for distinguishing between single-step and multi-step tasks is whether there is multi-step combined circuit reasoning in human analysis. The criteria for distinguishing simple from complex tasks are whether the total components of the circuit exceed 10 or whether there are more than five components with over two pins (e.g., transistors, operational amplifiers); meeting either criterion groups it into complex tasks. Additionally, if a task requires more than four steps of combined-circuit reasoning (e.g., small-signal model equivalence), it is grouped into complex multi-step tasks. The number of tasks in each category is shown in Fig. 4.

5 Experiments

5.1 Experiment details

5.1.1 Evaluating LLMs

Our evaluation covers nine leading LLMs, which are mainly composed of advanced closed-source commercial LLMs. We evaluate the analog circuit reasoning capacity of GPT4o, GPT4.1, Deepseek-V3, Gemini-2.0-Flash, Qwen3-32B, Qwen2.5-Max, Llama4-17B-128E, Doubao-Seed-1.6, and Hunyuan-Turbos. All LLMs do not activate thinking modes if equipped with them.

5.1.2 Metrics

We use $Pass@k$[43] $(k = 1,5)$ and $R@k$ $(k = 5)$ to evaluate the performance of LLMs on ACR-Bench. Pass@k is a universal metric for complex tasks, defined as the probability of at least one success in k trials. It is computed by $Pass@k = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}$ where n represents the total number of trials and c is the number of successful trials. Similarly, we designed the $R@k$ metric, defined as the probability of achieving all correct results in k trials. It is computed by $R@k = \frac{\binom{c}{k}}{\binom{n}{k}}$. If an LLM produces entirely correct outputs across k trials for a task, it is considered consistently reliable on that task. $R@k$ demonstrates LLMs' capability to independently carry out the analog circuit reasoning task successfully without human supervision. It indicates the feasibility of utilizing LLMs for fully automated analog circuit design based on their reasoning result. Both selected metrics are theoretically unbiased estimators, and the derivation for Pass@k is provided in previous work by Chen et al.[43]. The derivation for R@k is provided in Appendix A3.

5.1.3 Baseline

Currently, existing research has not yet developed solutions specifically designed for analog circuit reasoning with large language models. We select the circuit analysis and reasoning workflow within the following design frameworks as the baseline for comparison with our method: (1) Analogxpert[13]: using its sub-circuit library for real-world tasks and a designed reasoning method. (2) SizingAgent[15]: using its planning and execution strategy for circuit analysis. (3) Analogcoder[8]: using its incorporating subcircuits lists and one-shot CoT prompts. (4) Atelier[9]: using its supplementary knowledge and prompts designed for analyzing and designing circuits. (5) AMS-KG[14]: using its specifically designed reasoning method.

5.1.4 Settings

For all LLMs, we performed 10 tests for each task $(n = 10)$. When directly evaluating LLMs, we employ 0-shot prompts to eliminate the effects of few-shot prompt sensitivity[44]. To balance the stability and randomness of the results[45], we set all LLMs' temperature to 0.5. To compare whether the output expressions of LLMs are truly equal to human results, we introduce the Sympy parsing package to parse the expressions[46], which can simplify the expression and compare mathematical equality. In our prompt, we specify that the output answer must be enclosed between the designated tags `<answer>` and `<answer/>`. Only the string content inside these tags shall be extracted during expression extraction. If equal signs are present, expressions following the last equal ones are compared. Otherwise, the two expressions are compared directly.

5.2 Results

5.2.1 Overall performance

Table 3 compares our ACR-Agent directly using base models and baselines. For each improvement method, experiments were conducted using the advanced GPT-4.1 and DeepSeek-V3 as base models. The ACR-Agent successfully improved the *Pass*@1, *Pass*@5, and *R*@5 metrics of both models on the ACR-Bench, outperforming the baseline methods. A comparison of the results for GPT-4.1 and DeepSeek-V3 reveals that the stronger an LLM's inherent analog circuit reasoning capability, the greater the performance improvement achieved by our enhancement method. Furthermore, we found that among the five selected baselines, none effectively enhanced circuit structure–performance coupled reasoning capability; some even exhibited degradation, which probably limits the overall circuit design capabilities of their frameworks.

5.2.2 Base LLM evaluation

We directly tested the performance of different LLMs on ACR-Bench. ACR-Bench successfully evaluates and differentiates the analog circuit reasoning capabilities of various LLMs. Results are shown in Fig. 6. DeepSeek-V3 demonstrated the best performance, achieving *Pass*@1 of 0.36. However, even the most advanced LLMs currently achieve only around 0.5 in *Pass*@5, indicating limited reasoning capabilities for analog circuits. Their *R*@5 metrics fall below 0.2, suggesting low reliability for most task analysis. These results show that while current LLMs hold value as supervised reasoning assistants for analog circuit design, their low consistency makes it difficult to achieve highly automated analog circuit design workflows. Detailed results of base LLM evaluation are shown in Appendix A4.

5.2.3 Ablation study

We evaluated the effectiveness of components within ACR-Agent using the DeepSeek-V3 model, with results presented in Table 4. We conducted ablation studies on the Choosing Agent, its Concept Library, its Subcircuit Library, the Pre-analysis Agent's planning LLM (abbreviated as plan in Table 4) and splitting LLM (abbreviated as split in Table 4), and its constraints to evaluate their individual impacts on the ACR-Agent framework in analog circuit reasoning tasks. The results show that the existence of each component can impact the results, and the complete ACR-Agent framework achieves overall best performance on the whole ACR-Bench. *Basic Libraries* corresponds to the experimental setting that directly inputs all entries in libraries to the Analysis Agent, without any other agent. Its result shows that the ACR-Agent framework amplifies the positive effect of knowledge entries. *w/o Libraries* corresponds to the experimental setting without any knowledge libraries, and this result demonstrates that the multi-stage reasoning framework itself yields a limited yet positive impact,

Table 3. Main results. The performance of ACR-Agent on ACR-Bench.

Method	ACR-Bench			Simple single-step			Simple multi-step			Complex single-step			Complex multi-step		
	Pass@1	Pass@5	R@5	Pass@1	Pass@5	R@5	Pass@1	Pass@5	R@5	Pass@1	Pass@5	R@5	Pass@1	Pass@5	R@5
GPT4.1	0.30	0.48	0.14	0.48	0.70	0.24	0.24	0.45	0.08	0.18	0.35	0.05	0.09	0.11	0.03
Analogxpert (GPT4.1)	0.30	0.51	0.12	0.44	0.68	0.21	**0.27**	0.48	0.04	0.23	0.45	0.05	0.09	0.17	0.05
SizingAgent (GPT4.1)	0.29	0.50	0.11	0.41	0.66	0.19	0.26	**0.49**	0.07	**0.24**	0.44	0.06	0.07	0.15	0.01
Analogcoder (GPT4.1)	0.28	0.45	0.14	0.43	0.63	0.26	0.23	0.43	0.05	0.16	0.34	0.05	0.10	0.13	0.06
Atelier (GPT4.1)	0.23	0.41	0.08	0.33	0.54	0.14	0.22	0.40	0.09	0.14	0.36	0.01	0.07	0.13	0.01
AMS-KG (GPT4.1)	0.30	0.49	0.13	0.44	0.67	0.21	**0.27**	0.46	**0.11**	0.18	0.40	0.05	0.10	0.17	0.02
ACR-Agent (GPT4.1)	**0.37**	**0.56**	**0.18**	**0.60**	**0.81**	**0.34**	**0.27**	0.41	0.08	0.22	**0.47**	**0.07**	**0.12**	**0.20**	0.02
DeepSeek-V3	0.36	0.53	0.19	0.54	0.71	0.34	0.24	0.44	0.13	0.25	0.52	0.06	0.15	0.23	0.06
Analogxpert (DeepSeek-V3)	0.32	0.51	0.18	0.47	0.69	0.28	0.22	0.36	0.13	0.27	0.53	0.11	0.11	0.21	0.05
SizingAgent (DeepSeek-V3)	0.34	0.56	0.16	0.53	0.79	0.29	0.20	0.42	0.05	0.26	0.52	0.11	0.10	0.19	0.03
Analogcoder (DeepSeek-V3)	0.34	0.52	0.19	0.52	0.73	0.34	0.27	**0.47**	0.13	0.20	0.39	0.07	0.11	0.20	0.02
Atelier (DeepSeek-V3)	0.31	0.50	0.18	0.46	0.67	0.30	0.25	0.44	**0.14**	0.23	0.45	0.10	0.07	0.18	0.00
AMS-KG (DeepSeek-V3)	0.37	0.53	0.21	0.55	0.71	0.36	0.26	0.38	**0.14**	0.31	0.59	**0.13**	0.10	0.22	0.01
ACR-Agent (DeepSeek-V3)	**0.44**	**0.61**	**0.26**	**0.68**	**0.86**	**0.46**	**0.28**	0.37	0.13	**0.33**	**0.60**	**0.13**	**0.16**	**0.28**	**0.06**

Pass@1: probability of success in one trial. Pass@5: probability of at least one success among five repeated trials. R@5: probability of success across all five repeated trials. The bold values indicate the best-performance metrics of different methods built upon the same base model on ACR-Bench and various tasks.

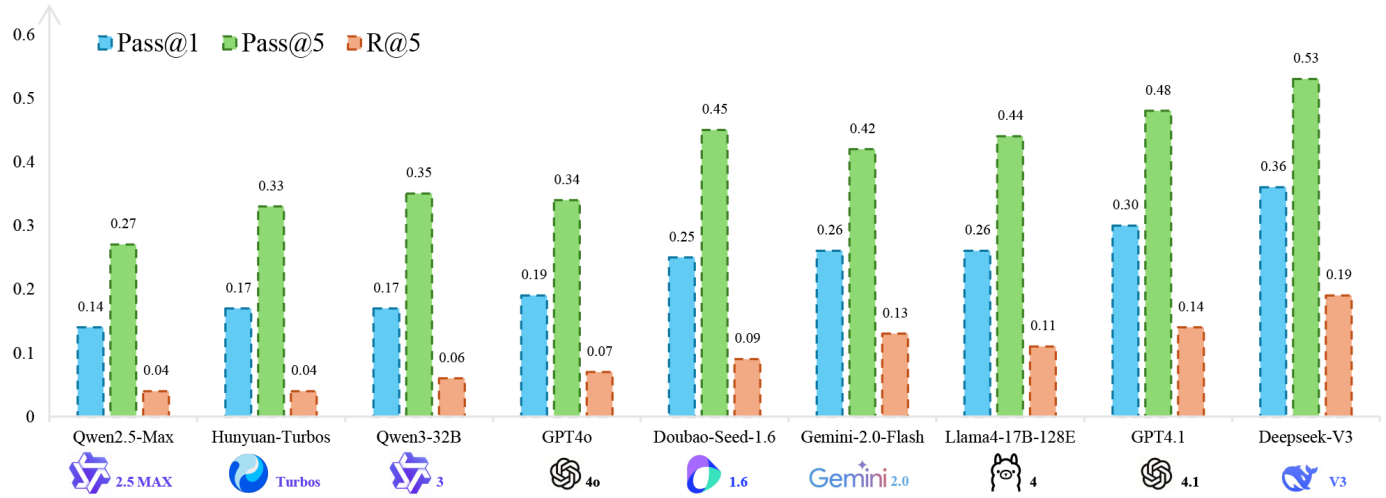

Fig. 6 The performance of different LLMs on ACR-Bench. LLMs are ranked by their *Pass*@1.

https://doi.org/10.48130/ker-0026-0012

which is particularly concentrated on complex single-step tasks. *w/o Choosing Agent* corresponds to the experimental setting that removes the process of choosing knowledge and inputs all entries in the two libraries to the following Pre-analysis Agent and Analysis Agent, and its result shows that the choosing agent itself has the ability to choose proper knowledge entry, thus improving the performance.

We further analyze the task-dependent gains of each module on analog circuit reasoning. The Planning LLM benefits simple multi-step tasks, where step-by-step decomposition helps the model follow correct small-signal or DC derivation chains. However, it slightly degrades performance on complex single-step tasks, as over-segmented subtasks may distract from the input knowledge and introduce minor noise. The Splitting LLM benefits complex single-step tasks by grouping devices into functional blocks. However, it shows a slight drop in performance on simple single-step tasks. To more precisely test the influence of the Planning LLM and the Splitting LLM, we have conducted manual error analysis on these two ablation settings. The Concept Library and Subcircuit Library mainly benefit tasks relying on principle consistency and classic circuit adaptation, which are exactly the two key challenges in analog circuit reasoning. All modules collectively yield the best performance on the full reasoning benchmark, and task-specific noise is limited to narrow categories rather than widespread degradation. In an EDA application, if the directly or indirectly targeted circuit analysis tasks mainly fall in one specific category, adjusting the framework to respond well correspondingly is a good choice.

5.2.4 Manual error analysis

We manually reviewed 604 incorrect answers from two runs of DeepSeek-V3, ablation setting w/o Pre-analysis Agent (plan), ablation setting w/o Pre-analysis Agent (split), and full ACR-Agent. Errors were classified as topology, concept, generalization, approximation, or expression (Fig. 7). Topology errors reflect an incorrect circuit structure, concept errors show wrong definitions of parts or parameters, generalization errors appear when correct ideas are not applied to new topologies or calculations, approximation errors replace given symbols with imprecise estimates, and expression errors invent parameters that were not provided.

More specifically, the criterion for topology errors is the presence of explicit textual mistakes in describing the topological relationships within the output. The criterion for concept errors is the output of incorrect theoretical formulas when defining certain parameters, where such definitions involve pure conceptual misunderstandings independent of the task-specific symbols. The criterion for generalization errors is errors caused by flawed procedural reasoning in the absence of concept errors, including mismatches between formulas and task parameters, mathematical reasoning mistakes, and similar issues. The criterion for approximation errors is the unjustified omission of certain parameters either before or after the reasoning process. The criterion for expression errors is the introduction of parameters that are not provided in the task statement. All borderline cases that are uncertain for classification by a single operator are classified following discussion by two experienced human operators.

Most DeepSeek-V3 mistakes fell into concept and generalization categories, indicating weak coupling between concept and reasoning and little generation on variant circuits, which is closely related to shortcomings in circuit design. ACR Agent cuts these two types sharply and thus improved its ACR-Bench score. The Splitting LLM (w/o Pre-analysis Agent [plan]) generates more errors in generalization capability and expression classification. This is probably because functional area segmentation breaks intact DC and AC circuits, hindering the model's analysis of large-scale circuits. Furthermore, the

Table 4. Results of ablation study.

Method	ACR-Bench			Simple Single-step			Simple Multi-step			Complex Single-step			Complex Multi-step		
	Pass@1	Pass@5	R@5	Pass@1	Pass@5	R@5	Pass@1	Pass@5	R@5	Pass@1	Pass@5	R@5	Pass@1	Pass@5	R@5
DeepSeek-V3	0.36	0.53	0.19	0.54	0.71	0.34	0.24	0.44	0.13	0.25	0.52	0.06	0.15	0.23	**0.06**
Basic libraries	0.40	0.57	0.25	0.65	0.81	**0.46**	0.23	0.38	0.13	0.27	0.53	0.08	0.11	0.23	0.01
w/o Libraries	0.37	0.57	0.20	0.57	0.80	0.32	0.24	0.36	0.15	0.29	0.58	0.09	0.14	0.25	0.04
w/o Subcircuit Library	0.39	0.58	0.22	0.59	0.82	0.36	0.27	0.41	**0.17**	0.28	0.54	0.11	0.13	0.23	0.03
w/o Concept Library	0.41	0.60	0.21	0.64	**0.86**	0.39	0.27	0.43	0.12	0.32	0.53	0.10	0.10	0.22	0.00
w/o Choosing Agent	0.41	0.57	0.24	0.64	0.80	0.43	0.25	0.35	0.14	0.30	**0.60**	0.10	0.13	0.23	0.03
w/o Constraints	0.41	**0.61**	0.21	0.65	**0.86**	0.39	0.27	**0.45**	0.12	**0.33**	0.56	0.10	0.10	0.24	0.00
w/o Pre-analysis Agent (split)	0.42	0.58	0.24	0.65	0.84	0.41	**0.30**	**0.45**	0.16	0.25	0.48	0.11	0.14	0.23	0.03
w/o Pre-analysis Agent (plan)	0.42	0.60	0.24	0.66	**0.86**	0.43	0.25	0.35	0.13	0.32	0.56	**0.17**	0.15	**0.28**	0.02
ACR-Agent	**0.44**	**0.61**	**0.26**	**0.68**	**0.86**	**0.46**	0.28	0.37	0.13	**0.33**	**0.60**	0.13	**0.16**	**0.28**	**0.06**

Pass@1: probability of success in one trial. Pass@5: probability of at least one success among five repeated trials. R@5: probability of success across all five repeated trials. The bold values indicate the best-performance metrics of different methods built upon the same base model on ACR-Bench and various tasks.

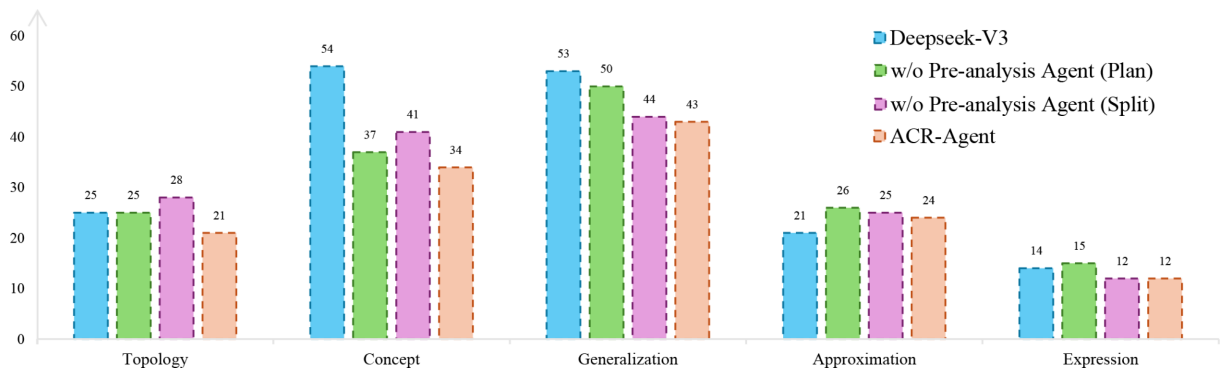

Fig. 7 Results of manual error analysis.

splitting agent occasionally modifies the parameter notations defined in the tasks during output, triggering expression errors. The Planning LLM (w/o Pre-analysis Agent [split]) generates more errors in topological structure and conceptual classification. This phenomenon may stem from overly refined task settings, which prevent the model from effectively utilizing the concept library and impair the recognition of topological paths.

5.2.5 Computational cost analysis

We conducted a computational cost analysis to quantify the additional token overhead introduced by the ACR-Agent. The results are shown in Table 5. All experiments are repeated on the full ACR-Bench dataset, and we report the average per-task consumption metrics. The Input Tokens and Output Tokens are directly extracted from the API responses, representing the total number of input and output tokens consumed per task. The Cache Hit Tokens are also extracted from the API responses, corresponding to the number of input tokens served by the input in-cache acceleration. The cached context and prompts are built upon the previous inputs. To eliminate cross-contamination of cached tokens between different test rounds, we use distinct DeepSeek APIs for each round of evaluation. The cost is calculated based on the actual monetary amount charged by the API service. As shown in the table, ACR-Agent significantly increases the total token consumption. However, with the structured prompt design and fixed framework, the cache hit rate increases from 26% to 62%, which effectively mitigates the overhead and results in a moderate rise in the total per-task cost. Notably, the average cost of ACR-Agent remains well below US $0.01 per task, demonstrating acceptable economic efficiency. Given the substantial improvements in canonical circuit reasoning accuracy and consistency, such a moderate increase in cost is considered a reasonable trade-off.

Table 5. Token consumption of ACR-Agent on ACR-Bench.

Method	Input tokens	Cache hit tokens	Output tokens	Cost ($)
DeepSeek-V3	614	158	530	0.0009
ACR-Agent	8,174	5,072	1,513	0.0033

5.2.6 Case study of analog circuit reasoning

Four representative successful and failed analog circuit reasoning tasks are shown in Fig. 8. We selected one representative example for each of the four problem classifications where ACR-Agent achieved clear performance improvements over the single-model baseline solution. We compare the results of DeepSeek-V3 and its agent-enhanced variant ACR-Agent and found that ACR-Agent successfully overcomes the previous shortcomings in structure–performance coupled reasoning:

1. Figure 8a shows a simple single-step task of non-linear circuits. In its reasoning, DeepSeek-V3 processed nodes sequentially, incorrectly determining that transistor Q_1 was partially shorted, leading to $\frac{R}{R_1}$ and 0. In ACR-Agent, the Choosing Agent successfully selected the subcircuit entry 'Transistor Voltage Regulator', and the Analysis Agent correctly identified that transistor Q_1 was non-operational. Combining this with the properties of the operational amplifier, the Analysis Agent derived the result $-\frac{2R}{R_1}$.

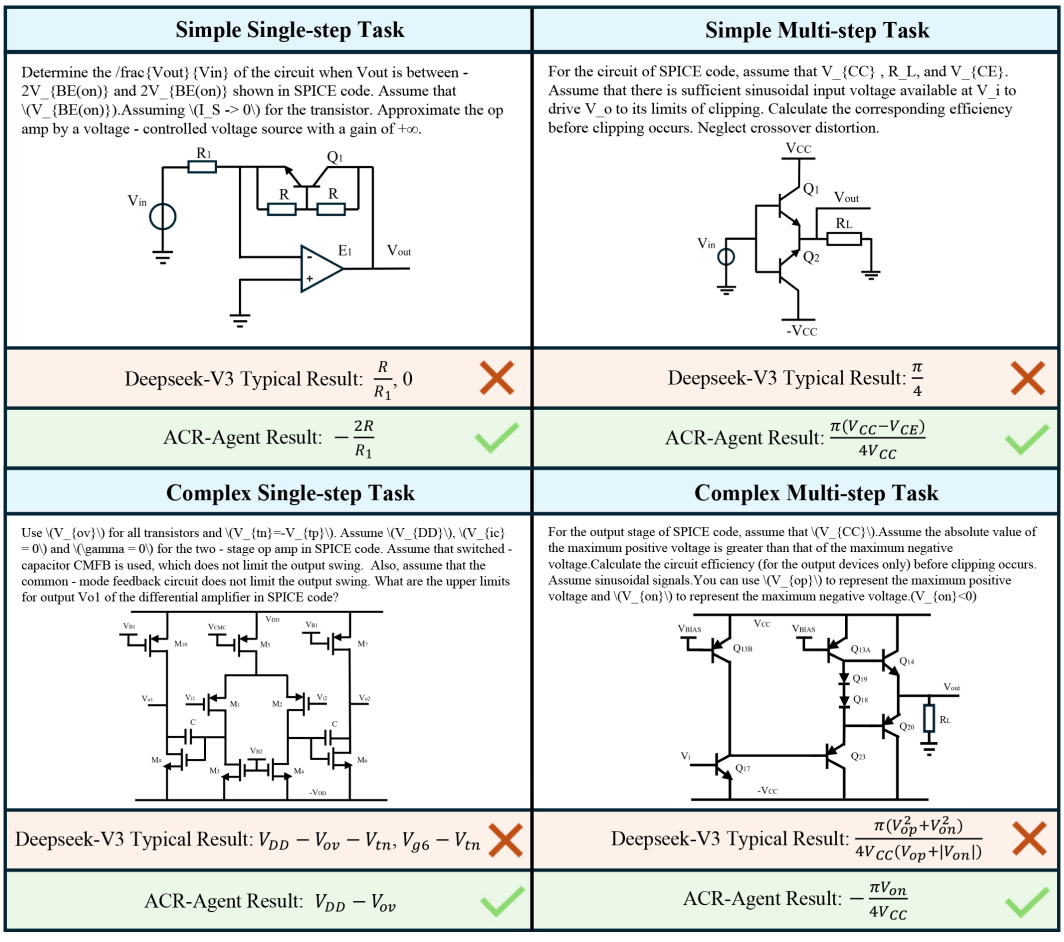

Fig. 8 Four representative tasks of successful reasoning and errors. All circuits in tasks are SPICE Netlist, which is simplified by circuit diagrams in the figure.

https://doi.org/10.48130/ker-0026-0012

2. Figure 8b shows a single multi-step task of output stages. In its reasoning, DeepSeek-V3 incorporates its background knowledge and directly disregards the given V_{CE}, incorrectly arriving at $\frac{\pi}{4}$, which is an example of limited generalization ability. In ACR-Agent, the planning LLM in Pre-analysis Agent successfully generates a subtask for output voltage analysis. The Analysis Agent first reasons to obtain the voltage amplitude V_{out}, and then, based on this result, reasons to the correct expression $\frac{\pi(V_{CC}-V_{CE})}{4V_{CC}}$.

3. Figure 8c shows a complex single-step task of full operational amplifiers. In its reasoning, DeepSeek-V3 fails to find a DC path directly related to the output in its analysis; it outputs that both M_7 and M_6 are relevant to the upper voltage limit, and therefore incorrectly arrives at $V_{DD}-V_{ov}-V_{tn}$. It also introduces self-defined variables to the final result during its reasoning, such as V_{g6}. In ACR-Agent, the Pre-analysis Agent's splitting LLM succeeds in extracting the correct DC path and listing the correct devices in it. The Analysis Agent then reasons out the voltage of each node in the DC path following the guidance of subtasks and the correct expression $V_{DD}-V_{ov}$ step by step.

4. Figure 8d shows a complex multi-step task of output stages. In its reasoning, DeepSeek-V3 understands the concept of clipping without examining the sinusoidal symmetry of the input signal, leading it to a wrong output-signal amplitude of $V_{op}+|V_{on}|$ and an efficiency of $\frac{\pi(V_{op}^2+V_{on}^2)}{4V_{CC}(V_{op}+|V_{on}|)}$, which is a typical example of representation–reasoning decoupling. In ACR-Agent, the Choosing Agent successfully selects the concept entry 'Power in Circuit', and the Analysis Agent accurately reasons out the correct amplitude $-2V_{on}$, then reasons out the result $-\frac{\pi V_{on}}{4V_{CC}}$ based on it.

Table 6. ACR-Agent application in circuit designing on the Analogcoder benchmark.

Model	Metrics	Analogcoder (w/o tools)		Analogcoder	
		Base	ACR-Agent	Base	ACR-Agent
GPT4.1	Pass@1	0.57	**0.59**	0.69	**0.72**
	Pass@5	0.68	**0.70**	0.90	**0.96**
DeepSeek-V3	Pass@1	0.54	**0.60**	0.63	**0.69**
	Pass@5	0.62	**0.71**	0.82	**0.88**

5.2.7 ACR-Agent in analog circuit design

To verify that our ACR-Agent can effectively integrate with analog circuit design frameworks, we incorporated the ACR-Agent framework into Analogcoder. We then evaluated the circuit design capabilities of the improved Analogcoder with the open-source Analogcoder benchmark, which contains 24 analog circuit design tasks with different themes and difficulty levels, along with corresponding functional verification. Using GPT-4.1 and DeepSeek-V3 as the base LLM, we compare the performance of ACR-Agent-improved and unimproved Analogcoder (Retry = 3) with and without utilizing subcircuit tools. As shown in Table 6, the enhanced base LLMs within the ACR-Agent framework demonstrate significant improvements on the Analogcoder Benchmark, both with and without utilizing subcircuit tools provided in Analogcoder. This shows that ACR-Agent with improved analog circuit structure–performance coupled reasoning can be effectively applied to enhance the performance of analog circuit design tasks. However, since physical layout and silicon implementation are not included in the current validation, the current experiment only shows its potential to future practical use. Detailed results of ACR-Agent in analog circuit design are shown in Appendix A5.

5.2.8 Case study of analog circuit design

Four representative designs of DeepSeek-V3 and its agent-enhanced variant ACR-Agent in Analogcoder, utilizing subcircuit tools, are shown in Fig. 9. We selected two representative tasks that best demonstrate the performance improvements achieved by integrating ACR-Agent into Analogcoder. For the four selected analog circuit designs, in addition to conducting simple functional verification using Analogcoder's built-in PySpice-based verification tool, we manually stimulated and validated the circuit effectiveness on SMIC18mmRF after simple voltage scaling. SMIC18mmRF is a process library developed by Semiconductor Manufacturing International Corporation (SMIC) for the 180 nm RF CMOS process, commonly used in the design and

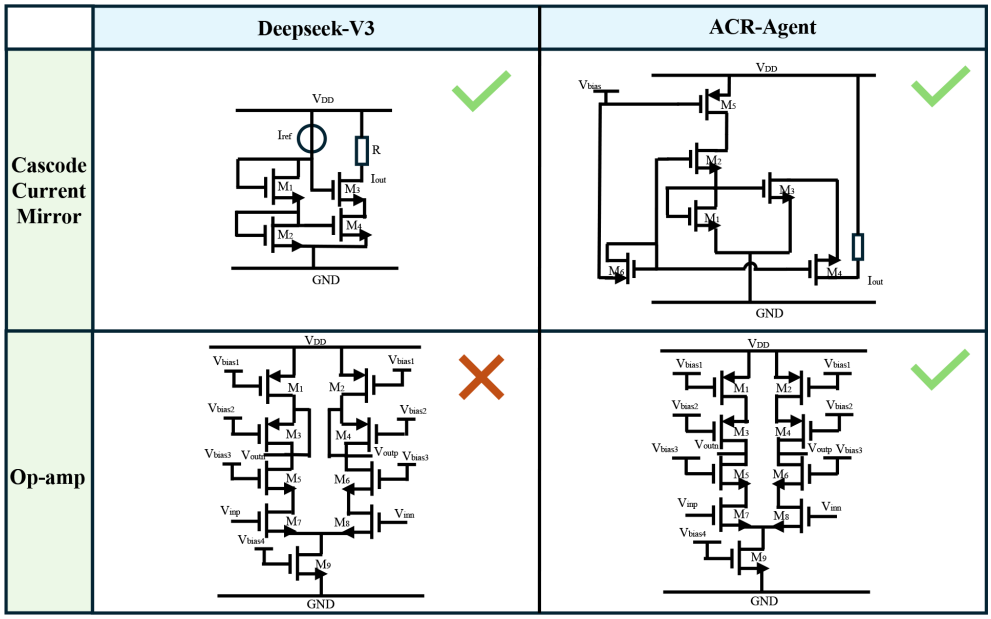

Fig. 9 Four representative design of DeepSeek-V3 and ACR-Agent based on DeepSeek-V3 in Analogcoder with utilizing subcircuit tools. All circuits in their design are PySpice code, which is simplified by circuit diagrams in the figure.

simulation of analog circuits, especially RF circuits. In this part, all circuits marked success have passed the performance verification on SMIC18mmRF, demonstrating their preliminary practical potential for circuit design. We compare the designs of DeepSeek-V3 and its agent-enhanced variant ACR-Agent in Analogcoder and find that the circuits designed by ACR-Agent are more practical and better able to handle complex circuit topologies:

1. The Cascode Current Mirror section presents two successful stable current sources based on current mirror designs. Even though clearly requested in Analogcoder, DeepSeek-V3 fails to output any reasoning text before outputting Spice code. It designs the same ideal Cascode Current Mirror in every successful trial. The stable operation of the circuit relies on a stable fixed voltage bias, and it fails to work stably when the process variation of the actual chip or the load fluctuation is excessive. In addition, the designed circuit includes an ideal current source. The design of ACR-Agent is more practical and diverse with its accurate knowledge and serious circuit splitting. It utilizes M_6 to generate a PMOS bias aligned with NMOS, which adapts to the chip process and temperature, thereby improving the stability of the chip. It also replaces the ideal current source with a PMOS active current source, allowing current adjustment by modifying its parameters.

2. The Op-amp section presents one successful and one failed Telescopic Cascode Operational Amplifier design. DeepSeek-V3 in Analogcoder designs a failed Operational Amplifier after a long text of reasoning. In terms of feedback in the Analogcoder framework, DeepSeek-V3 also fails to deeply analyze its previously designed circuit and find the mistakes, but generates a new Op-amp from the very beginning. Thus, in the final output of Analogcoder, it incorrectly connects the sources and drains of M_3 and M_4, causing these two PMOS transistors to malfunction, ultimately resulting in the failure of the entire circuit. In contrast, ACR-Agent splits the targeted circuit into several parts, provides reasonable guidance for topology generation, and accurately handles the topology of nine MOS transistors, designing a combination of a current mirror and an amplifier. It also precisely configures four bias voltages, enabling the operational amplifier to function properly.

6 Conclusions

We advance large language models in canonical analog circuit analysis in analog circuit design by strengthening their structure–performance coupled reasoning. Specifically, we propose **ACR-Agent**, a multi-agent framework that plans, separates, and solves circuit-reasoning tasks, and construct **ACR-Bench**, the first benchmark for rigorous evaluation, to test the effectiveness of ACR-Agent. Experiments show that ACR-Agent both outperforms established design baselines on ACR-Bench and boosts the effectiveness of existing circuit-design LLMs, showing promising transfer potential toward practical usage.

The current study is limited by the reliance on general-purpose foundation models, the increase in tokens consumed, and the scale of the dataset. Future work will explore post-training techniques to create a dedicated analog-design assistant, further decrease the computational consumption, and expand the range of circuit-reasoning tasks.

Author contributions

The authors confirm contribution to the paper as follows: study conception and design: Zhou J, Li H, Jiang H; data collection: Zhou J; analysis and interpretation of results: Zhou J, Li H, Zhou L, Jiang H; draft manuscript preparation: Zhou J, Li H; supervision: Li H, Jiang H. All authors reviewed the results and approved the final version of the manuscript.

Data availability

The data that support the findings of this study are available in the repository: https://github.com/cnzjq1/Analog-reasoning-circuit.

Acknowledgments

This study was funded by Key R&D Program of Hubei Province (Project No. JSCX202500180), and the Key R&D Program of Wuhan City (Project No. 2025050602030055).

Conflict of interest

The authors declare that they have no conflict of interest.

Supplementary information accompanies this paper online at: https://doi.org/10.48130/ker-0026-0012.

Dates

Received 19 March 2026; Revised 26 May 2026; Accepted 16 June 2026; Published online 25 August 2026

References

[1] Razavi B. 2000. *Design of analog CMOS integrated circuits*. New York: McGraw-Hill. 704 pp https://dl.acm.org/doi/book/10.5555/1594009

[2] Oppenheim AV., Willsky AS, Nawab SH. 1997. *Signals & systems*. 2nd Edition. London: Pearson Education. 957 pp https://dl.acm.org/doi/book/10.5555/248702

[3] Gray PR, Hurst P, Lewis SH, Meyer RG. 2009. *Analysis and design of analog integrated circuits*. 5th Edition. Hoboken: John Wiley & Sons. 896 pp

[4] Tlelo-Cuautle E. 2013. *Integrated circuits for analog signal processing*. Berlin: Springer. 322 pp doi: 10.1007/978-1-4614-1383-7

[5] Liu M, Ene TD, Kirby R, Cheng C, Pinckney N, et al. 2023. ChipNeMo: domain-adapted llms for chip design. *arXiv* Preprint: 2311.00176

[6] Zhong R, Du X, Kai S, Tang Z, Xu S, et al. 2023. LLM4EDA: emerging progress in large language models for electronic design automation. *arXiv* Preprint: 2401.12224

[7] Vladimirescu A. 1994. *The SPICE book*. Hoboken: John Wiley & Sons. 412 pp https://dl.acm.org/doi/10.5555/528264

[8] Lai Y, Lee S, Chen G, Poddar S, Hu M, et al. 2025. Analogcoder: analog circuit design via training-free code generation. *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence, Philadelphia, 2025*. Menlo Park: AAAI. pp. 379−387 doi: 10.1609/aaai.v39i1.32016

[9] Shen J, Chen Z, Zhuang J, Huang J, Yang F, et al. 2026. Atelier: an automated analog circuit design framework via multiple large language model-based agents. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 45(1):31−44

[10] Huynh N, Lin B. 2025. Large language models for code generation: a comprehensive survey of challenges, techniques, evaluation, and applications. *arXiv* Preprint: 2503.01245

[11] He Q, Zeng J, He Q, Liang J, Xiao Y. 2024. From complex to simple: enhancing multi-constraint complex instruction following ability of large language models. *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, 2024*. Stroudsburg: ACL. pp. 10864−10882 doi: 10.18653/v1/2024.findings-emnlp.637

[12] Wang H, Feng S, He T, Tan Z, Han X, et al. 2023. Can language models solve graph problems in natural language? *Advances in Neural Information Processing Systems 36 (NeurIPS 2023), New Orleans, 2023*. New York: Curran Associates. pp. 30840−30861 https://proceedings.neurips.

cc/paper_files/paper/2023/file/622afc4edf2824a1b6aaf5afe153fa93-Paper-Conference.pdf
[13] Zhang H, Sun S, Lin Y, Wang R, Bian J. 2025. AnalogXpert: automating analog topology synthesis by incorporating circuit design expertise into large language models. *2025 International Symposium of Electronics Design Automation (ISEDA), Hong Kong, 2025*. Piscataway: IEEE. pp. 772−777 doi: 10.1109/ISEDA65950.2025.11100627
[14] Shi Y, Tao Z, Gao Y, Zhou T, Chang C, et al. 2025. AMSnet-KG: a netlist dataset for LLM-based AMS circuit auto-design using knowledge graph RAG. *ACM Transactions on Design Automation of Electronic Systems*. https://doi.org/10.1145/3736166
[15] Liu C, Olowe EA, Chitnis D. 2025. LLM-based AI agent for sizing of analog and mixed signal circuit. *2025 23rd IEEE Interregional NEWCAS Conference (NEWCAS), Paris, France, 2025*. Piscataway, USA: IEEE. pp. 90−94 doi: 10.1109/NewCAS64648.2025.11107079
[16] Rashid R, Krishna K, George CP, Nambath N. 2024. Machine learning driven global optimisation framework for analog circuit design. *Microelectronics Journal* 151:106362
[17] Hammoud A, Goyal C, Pathen S, Dai A, Li A, et al. 2024. Human language to analog layout using Glayout layout automation framework. *Proceedings of the 2024 ACM/IEEE International Symposium on Machine Learning for CAD, Snowbird, 2024*. New York: ACM. pp. 1−7 doi: 10.1145/3670474.3685971
[18] Dong Z, Cao W, Zhang M, Tao D, Chen Y, et al. 2023. Cktgnn: circuit graph neural network for electronic design automation. *arXiv* Preprint: 2308.16406
[19] Vijayaraghavan P, Shi L, Degan E, Mukherjee V, Zhang X. 2025. AUTOCIRCUIT-RL: reinforcement learning-driven LLM for automated circuit topology generation. *Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada, 2025*. Cambridge MA: JMLR. pp. 61498−61512 https://proceedings.mlr.press/v267/vijayaraghavan25a.html
[20] Chang CC, Lin WH, Shen Y, Zhou G, Chen Y, et al. 2024. LaMAGIC: advanced circuit formulations for language-model-based topology generation for analog integrated circuits. *ACM Transactions on Design Automation of Electronic Systems*. 31(5):1−21
[21] Chang CC, Lin WH, Shen Y, Chen Y, Zhang X. 2025. LaMAGIC2: advanced circuit formulations for language model-based analog topology generation. *Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada, 2025*. vol. 267. Cambridge, USA: PMLR. pp. 7351−7360 . https://proceedings.mlr.press/v267/chang25b.html
[22] Gao J, Cao W, Yang J, Zhang X. 2025. AnalogGenie: a generative engine for automatic discovery of analog circuit topologies. *arXiv* Preprint: 2503.00205
[23] Chien E, Li M, Aportela A, Ding K, Jia S, et al. 2024. Opportunities and challenges of graph neural networks in electrical engineering. *Nature Reviews Electrical Engineering* 1(8):529−546
[24] Chaudhuri J, Thapar D, Chaudhuri A, Firouzi F, Chakrabarty K. 2025. SPICED+: syntactical bug pattern identification and correction of trojans in A/MS circuits using LLM-enhanced detection. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 33(4):1118−1131
[25] Liu C, Chen W, Peng A, Du Y, Du L, et al. 2024. AmpAgent: an LLM-based multi-agent system for multi-stage amplifier schematic design from literature for process and performance porting. *arXiv* Preprint: 2409.14739
[26] Skelic L, Xu Y, Cox M, Lu W, Yu T, et al. 2025. CIRCUIT: a benchmark for circuit interpretation and reasoning capabilities of LLMs. *arXiv* Preprint: 2502.07980
[27] Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems 33 (NeurIPS 2020), online, 2020*. New York: Curran Associates. pp. 1877−1901 https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf
[28] Gao J, Cao J, Bu R, Zhu N, Guan W, et al. 2025. Promoting knowledge base question answering by directing LLMs to generate task-relevant logical forms. *Proceedings of the AAAI Conference on Artificial Intelligence* 39(22):23914−23922
[29] Wei J, Wang X, Schuurmans D, Bosma M, Xia F, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems 35 (NeurIPS 2022), New Orleans, 2022*. New York: Curran Associates. pp. 24824-24837 https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf
[30] Besta M, Blach N, Kubicek A, Gerstenberger R, Podstawski M, et al. 2024. Graph of thoughts: solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence* 38(16):17682−17690
[31] Jin W, Zhao B, Yu H, Tao X, Yin R, et al. 2023. Improving embedded knowledge graph multi-hop question answering by introducing relational chain reasoning. *Data Mining and Knowledge Discovery* 37(1):255−288
[32] Yu H, Wen J, Zheng Z. 2025. CAMEL: cross-modality adaptive meta-learning for text-based person retrieval. *IEEE Transactions on Information Forensics and Security*. 20:4651−4663
[33] Kambhampati S. 2024. Can large language models reason and plan? *Annals of the New York Academy of Sciences* 1534(1):15−18
[34] Shojaee P, Mirzadeh I, Alizadeh K, Horton M, Bengio S, et al. 2025. The illusion of thinking: understanding the strengths and limitations of reasoning models via the lens of problem complexity. *arXiv* Preprint: 2506.06941
[35] Arslan M, Ghanem H, Munawar S, Cruz C. 2024. A survey on RAG with LLMs. *Procedia Computer Science* 246:3781−3790
[36] Zhou Z, Tao R, Zhu J, Luo Y, Wang Z, et al. 2024. Can language models perform robust reasoning in chain-of-thought prompting with noisy rationales? *Advances in Neural Information Processing Systems 37 (NeurIPS 2024), Vancouver, Canada, 2024*. New York: Curran Associates. pp. 123846−123910 doi: 10.52202/079017-3936
[37] Zhang C, Goh XD, Li D, Zhang H, Liu Y. 2025. Planning with multi-constraints via collaborative language agents. *Proceedings of the 31st International Conference on Computational Linguistics, Abu Dhabi, 2025*. Stroudsburg: ACL. pp. 10054−10082 https://aclanthology.org/2025.coling-main.672
[38] Bhandari J, Bhat V, He Y, Rahmani H, Garg S, et al. 2024. Masala-CHAI: a large-scale SPICE netlist dataset for analog circuits by harnessing AI. *arXiv* Preprint:2411.14299
[39] Tao Z, Shi Y, Huo Y, Ye R, Li Z, et al. 2024. AMSNet: netlist dataset for ams circuits. *2024 IEEE LLM Aided Design Workshop (LAD), San Jose, CA, USA, 2024*. Piscataway, USA: IEEE. pp. 1−5 doi: 10.1109/LAD62341.2024.10691781
[40] Shi Y, Zhang Z, Wang H, Tao Z, Li Z, et al. 2025. AMSbench: a comprehensive benchmark for evaluating MLLM capabilities in AMS circuits. *arXiv* Preprint: 2505.24138
[41] Li M, Zhong J, Chen T, Lai Y, Psounis K. 2025. EEE-bench: a comprehensive multimodal electrical and electronics engineering benchmark. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 2025*. Piscataway, USA: IEEE. pp. 13337−13349 doi: 10.1109/CVPR52734.2025.01245
[42] Jiang M, Liu KZ, Zhong M, Schaeffer R, Ouyang S, et al. 2024. Investigating data contamination for pre-training language models. *arXiv* Preprint: 2401.06059
[43] Chen M, Tworek J, Jun H, Yuan Q, de Oliveira Pinto HP, et al. 2021. Evaluating large language models trained on code. *arXiv* Preprint: 2107.03374
[44] Ma H, Zhang C, Bian Y, Liu L, Zhang Z, et al. 2023. Fairness-guided few-shot prompting for large language models. *Advances in Neural Information Processing Systems 36 (NeurIPS 2023), New Orleans, 2023*. New York: Curran Associates. pp. 43136−43155 https://proceedings.neurips.cc/paper_files/paper/2023/file/8678da90126aa58326b2fc0254b33a8c-Paper-Conference.pdf
[45] Renze, M. 2024. The effect of sampling temperature on problem solving in large language models. *Findings of the association for computational linguistics: EMNLP 2024, Miami, Florida, USA, 2024*. Stroudsburg: ACL. pp. 7346−7356 doi: 10.18653/v1/2024.findings-emnlp.432
[46] Meurer A, Smith CP, Paprocki M, Čertík O, Kirpichev SB, et al. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science* 3:e103