TopoSizing: An LLM-aided Framework of Topology-based Understanding and Sizing for AMS Circuits

Ziming Wei[1], Zichen Kong[2], Yuan Wang[2], David Z. Pan[3*], Xiyuan Tang[4*]
[1]School of EECS, Peking University, Beijing, China
[2]School of Integrated Circuits, Peking University, Beijing, China
[3]Department of Electrical and Computer Engineering, The University of Texas at Austin, USA
[4]Institute for Artificial Intelligence, Peking University, Beijing, China
*Corresponding authors: dpan@ece.utexas.edu, xtang@pku.edu.cn

***Abstract*—Analog and mixed-signal (AMS) circuit design remains challenging due to the shortage of high-quality data and the difficulty of embedding domain knowledge into automated flows. Traditional black-box optimization achieves sampling efficiency but lacks circuit understanding, which often causes evaluations to be wasted in low-value regions of the design space. In contrast, learning-based methods embed structural knowledge but are case-specific and costly to retrain. Recent attempts with large language models (LLMs) show potential, yet they often rely on manual intervention, limiting generality and transparency. We propose TopoSizing, an end-to-end framework that performs robust circuit understanding directly from raw netlists and translates this knowledge into optimization gains. Our approach first applies graph algorithms to organize circuits into a hierarchical device–module–stage representation. LLM agents then execute an iterative hypothesis–verification–refinement loop with built-in consistency checks, producing explicit annotations. Verified insights are integrated into Bayesian optimization (BO) through LLM-guided initial sampling and stagnation-triggered trust-region updates, improving efficiency while preserving feasibility. On four real-world AMS circuits (OTA, FC-OTA, SACMP, LDO) in a 55-nm CMOS process, TopoSizing achieves 100% correctness in circuit understanding and parameter assignment. In constraint-satisfaction tasks, it delivers 1.4×–4.8× higher sample efficiency and 1.2×–3.5× faster runtime compared with strong baselines, while requiring 2–4× fewer LLM calls than prior LLM-based frameworks. Ablation studies confirm that reliable circuit understanding is essential for these gains.**

***Index Terms*—Analog and mixed-signal (AMS) circuits, electronic design automation (EDA), large language model (LLM), Bayesian optimization (BO)**

I. Introduction

Analog and mixed-signal (AMS) integrated circuits, such as operational amplifiers, comparators, and voltage references, remain indispensable in modern electronic systems. However, their design continues to rely heavily on expert intervention and domain-specific experience, making the process labor intensive and difficult to scale. Among the various stages, device sizing plays a particularly decisive role, as it determines key performance metrics. Yet, sizing is notoriously challenging: the design space is high-dimensional due to numerous interdependent parameters, and performance evaluation requires computationally expensive SPICE-level simulations. Together, these factors render sizing a critical but highly complex stage, one that this work directly aims to address.

The most straightforward way to formulate sizing is as a black-box optimization problem, where device parameters are directly mapped to performance metrics. In this setting, heuristic algorithms [1] and Bayesian Optimization (BO) and its variants [2]–[4] have been widely applied, offering high sample efficiency and topology-agnostic applicability. However, black-box methods do not perform what expert designers routinely do, namely **circuit understanding**. Without the ability to interpret the structure and functionality of a circuit, these approaches cannot map topology into meaningful design constraints or guidance. As a result, they often fail to avoid wasting evaluations in unpromising regions that a human engineer would immediately exclude. This gap highlights the importance of circuit understanding as a necessary ingredient for practical automation.

Circuit understanding, in this context, refers to the ability to reason from structure, parameters, and connectivity of circuits to infer their implications on performance and to translate this reasoning into design constraints or guidelines. The pursuit of such understanding has motivated researchers to explore learning-based methods. Early approaches trained surrogate models that mapped design parameters to performance metrics, effectively embedding domain knowledge into the optimization loop [5]–[8]. Later efforts extended this idea with graph-based algorithms to explicitly encode circuit structure, capturing functional relationships between devices and modules [9]–[12]. While these methods enriched the representation of circuits, they remain limited in practice: surrogate models are typically case-specific and require retraining whenever the topology changes, while dataset generation through costly simulations often outweighs the savings they provide. Consequently, despite their conceptual appeal, these approaches cannot scale or deliver efficient design automation.

Up to this point, existing strategies reveal a fundamental dilemma: achieving topology generalizability, sampling efficiency, and embedded circuit understanding simultaneously has appeared nearly impossible. Naturally, the criteria for a

form of *circuit understanding* suitable for automated design can be summarized as follows: **(a) explicit preservation for reuse**, meaning the extracted knowledge is represented explicitly and stored in a form that can be reused across different design stages such as constraint formulation, testbench generation, or layout guidance; **(b) structural generality**, meaning the method can extend naturally to diverse circuit families and topologies without case-specific retraining; and **(c) transparency and verifiability**, meaning the intermediate roles of devices, modules, and stages are made explicit in a form that can be inspected, validated, and trusted.

However, achieving such circuit understanding is far from trivial: it requires a powerful model that can reliably interpret circuit topology, incorporate relevant domain knowledge, and correctly map this understanding into design requirements. Fortunately, the emergence of large-language models (LLMs) offers a promising path forward. Trained in vast and diverse corpora, LLMs possess broad prior knowledge that encompasses fundamental concepts in circuit theory and analog design. In addition, they exhibit a degree of reasoning capability, enabling them to interpret circuit connectivity, propose plausible sizing strategies, and support design-oriented inference. Moreover, their flexible input–output modalities make them easily extensible within existing workflows. Together, these characteristics suggest that LLMs could reconcile sample efficiency, topology transferability, and circuit knowledge integration within a single framework. This possibility has attracted increasing attention in the analog design automation community [13]–[16].

Motivated by these advantages, recent works have explored the use of LLMs in AMS electronic design automation (EDA). However, current approaches still suffer from significant limitations in scalability, generality, and transparency; a detailed review is provided in Section II. Together, the limitations of existing approaches can be traced to two main sources. The first lies in the unreliability of circuit understanding. LLMs are not naturally suited for reasoning over graph-structured data [17] such as circuits, and raw netlists pose particular difficulties. Unlike designer-authored netlists, which often contain implicit hints such as ordered indexing, hierarchical organization, or meaningful node names, raw netlists, especially those extracted automatically from images or heterogeneous sources, lack such structure. Their nodes typically do not have informative labels, their ordering is arbitrary, and functional modules are not delineated. Therefore, it is unrealistic to assume that an LLM can directly parse these netlists into the correct functional roles [18]. Achieving reliable understanding requires working strictly from topology-only information and deriving a robust, high-quality interpretation akin to how human engineers decompose circuits. The second difficulty lies in how to effectively leverage the attained circuit understanding to achieve tangible improvements in optimization efficiency. This requires first the choice of a principled backbone optimization algorithm and then the design of appropriate intervention mechanisms. However, existing LLM-based works lack consensus on what constitutes the most effective integration strategy, and the approaches adopted so far remain relatively coarse. For example, some studies bypass circuit knowledge entirely by invoking an optimization algorithm directly, while others let the LLM propose candidate sampling points or prune the search space based on the current optimization state. A more systematic methodology is needed to translate circuit understanding into design efficiency, along with comparative studies to evaluate different intervention strategies.

In response to the above challenges, we propose an end-to-end framework that spans the analog front-end design flow, taking a raw netlist as input and producing a sized circuit as output, with circuit understanding accomplished in the process and preserved as intermediate knowledge in textual or annotated form. The framework begins with raw netlists and performs robust circuit understanding based solely on topology, assisted by graph algorithms and LLMs. This circuit understanding is not only essential for guiding subsequent design tasks within our framework but also tangible in the sense that the extracted knowledge can be retained for potential reuse in other applications. On top of this, sizing serves both as a strengthened optimization stage and as a validation mechanism: the acceleration achieved in sizing demonstrates the practical benefits of circuit understanding, while the correctness of sizing outcomes in turn substantiates the reliability of the proposed circuit understanding process.

Our main contributions are summarized as follows:

- We present a unified end-to-end framework for analog front-end design that integrates LLMs into the workflow. Starting from raw netlists, the framework performs topology-based circuit understanding and leverages the extracted annotations to enhance downstream tasks, with the knowledge preserved for potential reuse.
- We propose a graph-assisted methodology for reliable circuit understanding. Circuit information is systematically extracted into a hierarchical representation across three levels–component, module, and stage–and the LLM executes iterative reasoning in a hypothesis-verification-refinement loop, improving robustness while ensuring interpretability.
- We introduce an efficient integration of LLM-based circuit understanding with BO. By incorporating LLM-guided initial sampling and stagnation-triggered trust-region updates, our approach improves BO efficiency through simple yet effective mechanisms.
- We validate the framework on four real-world circuit cases, where TopoSizing achieves **100% correctness** in both circuit understanding and parameter assignment. In constraint satisfaction tasks, our method delivers **1.4× to 4.8× higher sample efficiency** and **1.2× to 3.5× faster runtime** compared with traditional optimization baselines.

II. Preliminaries

A. AMS Circuit Understanding and Sizing

From the above discussion, we note that our entire workflow begins with establishing a reliable *circuit understanding*. To evaluate this crucial step, we introduce two indicators: (i) the *classification accuracy* of devices and modules, which reflects whether circuit components are assigned to the correct functional roles; and (ii) the *correctness of design-constraint or parameter assignment*, which measures whether the extracted circuit information can be faithfully translated into sizing-related guidance.

To further demonstrate the practical value of circuit understanding, we take AMS circuit sizing as a downstream task. AMS sizing refers to the process of tuning device-level parameters—such as transistor widths, lengths, or bias currents—so that the resulting circuit meets all performance specifications such as gain, bandwidth, power consumption, and stability. Successful sizing requires searching over a high-dimensional design space, and its efficiency can be significantly improved when guided by accurate circuit understanding. Thus, sizing serves both as a key design objective and as an auxiliary validation of the usefulness of our framework.

B. Problem Formulation for Sizing

AMS circuit sizing can be cast as a constrained optimization problem. For the feasibility scenario, the task is to identify a design $\mathbf{x} \in \mathcal{X}$ that meets all performance specifications:

$$\text{find } \mathbf{x} \in \mathcal{X} \quad \text{s.t. } F_i(\mathbf{x}) \geq C_i, \ \ \forall i \in \{1, \ldots, N_c\}, \tag{1}$$

where $F_i(\mathbf{x})$ is the i-th performance metric, C_i its specification, and N_c the number of constraints.

For the single-objective constrained optimization, one target metric $F_t(\mathbf{x})$ is maximized while ensuring feasibility of all other constraints:

$$\max_{\mathbf{x} \in \mathcal{X}} \ F_t(\mathbf{x}) \quad \text{s.t. } F_i(\mathbf{x}) \geq C_i, \ \ \forall i \neq t. \tag{2}$$

This formulation highlights the two experimental settings considered in our framework: (i) pure feasibility search for rapid constraint satisfaction, and (ii) performance-driven optimization under design constraints.

C. LLMs for AMS EDA

Existing approaches of utilizing LLMs in AMS EDA can be broadly divided into two categories.

The first line of work develops domain-specific LLMs tailored for AMS design tasks. These methods aim to incorporate circuit-specific knowledge into pretraining or fine-tuning, but they are constrained by the scarcity of large, high-quality datasets. Despite auxiliary efforts such as extracting netlists from schematics or images [19], [20], automatically generating sizing scripts [21], and constructing test benches [22], ensuring correctness in circuit interpretation and parameter allocation still requires substantial manual validation. This bottleneck limits both scalability and reliability.

The second line of work attempts to directly apply general-purpose LLMs within the design process. In the context of sizing, for example, retrieval-augmented generation (RAG) and multi-agent discussion have been explored to provide design guidance [16], [23]. Other studies let LLMs intervene in optimization directly [13], [14]. However, these approaches typically process raw netlists without additional structure, which restricts applicability to simple topologies. Moreover, their interpretations are rarely verified, leaving the process opaque and lacking transparency. Even in optimization integration, strategies remain coarse and inconsistent: some pipelines bypass model intervention entirely [16], while others only allow the LLM to propose a few candidate samples per iteration [13].

Nevertheless, these efforts highlight the larger potential of LLM-in-the-loop workflows: as commercial models continue to improve, such pipelines could, in principle, achieve even stronger design performance. The key challenge, however, lies in ensuring interpretability and reliability. In particular, existing approaches rarely verify whether the LLM's circuit understanding is correct or whether its design guidance is appropriate—both of which are crucial for trustworthy automation. This gap motivates our work, where we propose a framework that not only leverages LLMs for circuit understanding but also incorporates mechanisms to validate and refine their interpretations.

In summary, while these works demonstrate the potential of LLMs in AMS EDA, they also reveal clear limitations in scalability, generality, and verifiability. Addressing these gaps motivates the systematic framework proposed in this paper.

D. Circuit Representation: Requirements and Choices

A central question in enabling LLMs for AMS EDA is how to represent circuits in a form that is both **information-preserving** and **LLM-compatible**. On the one hand, the representation must preserve all relevant structural and parametric information, since any loss of connectivity or device attributes would compromise the fidelity of subsequent analysis. On the other hand, the representation must be interpretable by LLMs, whose pretraining distribution is dominated by natural language rather than specialized graph data or domain-specific syntaxes.

From the perspective of information completeness, electrical circuits are naturally graphs: devices and terminals can be modeled as nodes, while electrical connections form edges. Such graph-based representations are widely used in learning-based electronic design automation (EDA), particularly with graph neural networks (GNNs) [9], [10], [12]. They provide a lossless encoding of circuit topology and attributes. However, these representations are not well aligned with LLMs. Raw adjacency matrices or edge lists are high-dimensional, order-sensitive, and difficult to tokenize effectively; linearizing them introduces artificial ordering biases; and functional relationships such as differential pairs or current mirrors remain implicit [17]. Consequently, although graph representations

excel at preserving information, they do not naturally support LLM reasoning.

Another direction is to augment the raw netlist with **semantic annotations**. For instance, designers may explicitly mark functional groups. Such annotations improve interpretability but typically require substantial manual effort, making them difficult to scale across large design spaces. A related approach is to employ code-like domain-specific languages (DSLs) for circuit description. These languages introduce specialized syntax to represent modules and hierarchical structure in a more formal and programmable way, akin to hardware description languages [**?**]. While such DSLs can provide concise and structured descriptions, they rely on conventions that pretrained LLMs are not exposed to, and thus may not be easily understood without additional fine-tuning or task-specific adaptation. In summary, both directions have demonstrated effectiveness within their respective application scenarios, but neither fully satisfies the dual requirement of being information-encompassing and LLM-compatible. This gap motivates our approach, which aims to balance these trade-offs by transforming graph-extracted circuit information into structured natural-language descriptions that preserve essential topology while remaining aligned with LLMs' strengths.

III. Methodology

A. Overview

In this section, we describe the detailed implementation of **TopoSizing**, a fully automated framework that integrates graph-based processing with LLMs to achieve reliable circuit understanding and sample-efficient optimization (Fig. 1). The workflow is carried out in three stages. First, raw netlists are transformed into hierarchical graph representations that capture component, module, and stage-level structure. Based on this representation, LLM agents iteratively perform circuit understanding, producing explicit and verifiable annotations of functional roles. Finally, this circuit understanding is coupled with an adapted BO, where LLM guidance refines the design space and gives instructions on trust region update. By moving from structural encoding to functional interpretation and ultimately to optimization, TopoSizing integrates all stages into a coherent end-to-end pipeline.

B. Topological Information Extraction

The first step toward reliable circuit understanding is to choose an LLM- and algorithm-friendly circuit representation that is both expressive and easy to process. Therefore, we attempt to construct a structured container that captures the circuit at three hierarchical levels: *component*, *module*, and *stage*. This organization preserves both fine-grained details and higher-level functional context, making the representation both expressive and easy to query. An additional advantage is that hierarchical abstraction effectively reduces the connection complexity inherent in graph-structured netlists. Both advantages of this structured graph container are particularly beneficial in improving the robustness of LLM-based circuit understanding [24]. The design of this hierarchy is inspired by how analog designers analyze circuits: starting from the supply to divide the circuit into stages, then identifying common module sub-circuits to quickly infer functionality, and finally inspecting individual component connections to verify details.

*1) **Circuit to graph transformation**:* Circuits naturally exhibit the properties of a graph: devices and nets form nodes, and their connections define edges. Leveraging this intrinsic structure, we transform a raw netlist into a structured, semantically enriched graph in a fully automated way. This process is almost "free" since it reuses the connectivity already encoded in the netlist and produces a unique, lossless representation that preserves all original information.

$$
\begin{aligned}
\mathcal{G} &= (\mathcal{V}, \mathcal{E}),\\
\mathcal{V} &= \mathcal{V}_{\text{dev}} \cup \mathcal{V}_{\text{net}},\\
\mathcal{V}_{\text{dev}} &= \{\, v_i \mid \tau(v_i) \in \{\text{NMOS}, \text{PMOS}, R, C, I, V\} \,\},\\
\mathcal{V}_{\text{net}} &= \{\, v_j \mid \tau(v_j) \in \{\text{net}, \text{GND}, \text{VDD}\} \,\},\\
\mathcal{E} &\subseteq \{\, (v_i^{\text{dev}}, v_j^{\text{net}}, \ell) \mid \ell \in \mathcal{L} \,\},\\
\mathcal{L} &= \{G, S, D, R, C, I, V\}.
\end{aligned}
$$

Here, $\mathcal{G}$ is the circuit graph with node set $\mathcal{V}$ and edge set $\mathcal{E}$. The node set is partitioned into device nodes $\mathcal{V}_{\text{dev}}$ and net nodes $\mathcal{V}_{\text{net}}$, where $\tau(v_i)$ denotes the node type. Each edge $\varepsilon \in \mathcal{E}$ connects a device node to a net node with label $l \in \mathcal{L}$, where l indicates terminal roles or device types.

Edges $\varepsilon \in \mathcal{E}$ connect device nodes to net nodes with a label $\ell \in \mathcal{L}$. For MOSFETs, labels indicate terminal roles: Gate (G), Source (S), Drain (D). For other devices, the label corresponds to the device type. Since connections only occur between devices and nets, the resulting graph is bipartite, capturing both topology and port-level semantics.

This structured encoding offers two key benefits: (1) it is lossless, preserving every device, connection, and functional role, and (2) it is directly compatible with graph-based algorithms. As a result, it serves as an ideal foundation for subsequent topology analysis, module recognition, and optimization.

*2) **Sub-circuit Matching**:* To further simplify the circuit graph while retaining meaningful functional structure, we take advantage of a characteristic unique to analog circuits: the frequent reuse of well-established basic modules such as differential pairs, current mirrors, and cascode stages. This is where our graph-based representation shows its strength: once a circuit is represented as a graph, we can directly apply subgraph algorithms to identify and manipulate these recurring patterns.

In our approach, we first construct a library of commonly used analog building blocks. Each building block is a labeled subgraph in which both node types and edge types are explicitly specified, encoding canonical topological patterns. These templates not only guide the matching process, but also link specific transistors and devices to their functional roles within a module, enabling us to later reason about how design parameters influence performance. The initial library covers differential pairs, current mirrors, cascode stages, cascode

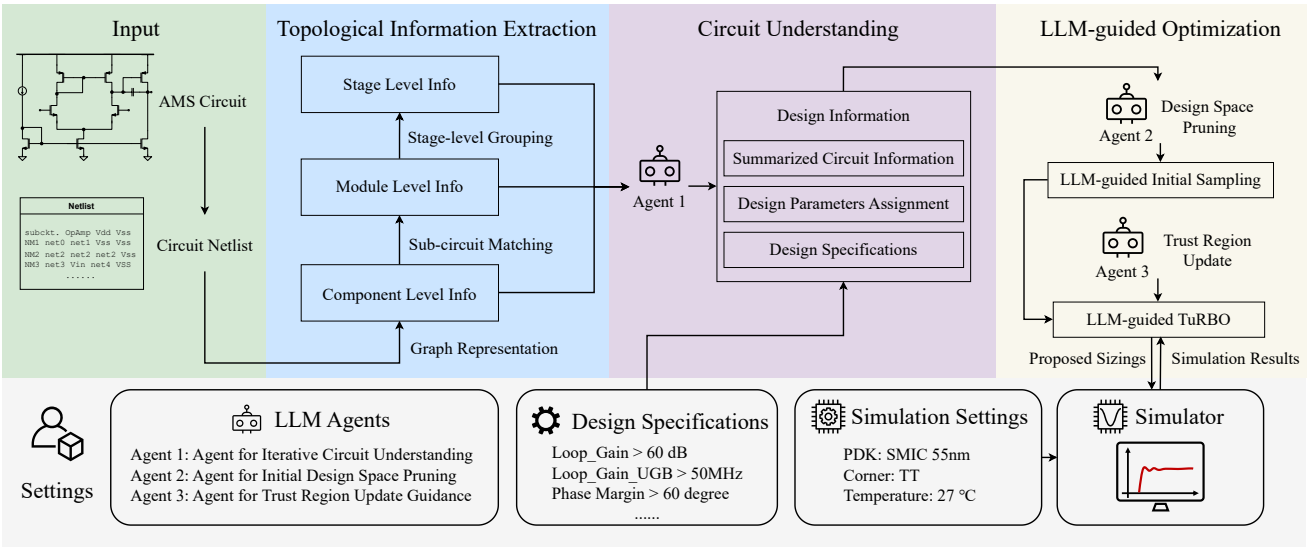


Fig. 1: The work flow of TopoSizing

current mirrors, Class-AB stages, and diode-connected MOS devices. These modules are chosen because they constitute the fundamental building blocks of analog design, are repeatedly reused across circuits, and together span almost all practical topologies [25], [26]. The library can also be readily extended to more complex structures.

Based on the graph representation of the circuit, we perform subgraph isomorphism to detect template matches. When a match is found, we replace the matched devices with a single supernode representing the identified module. Net nodes, which define the electrical interconnection points, are preserved to maintain global connectivity. This replacement does more than reduce complexity, it also embeds new semantic information into the graph. By renaming the supernode and adjusting the edge types at its boundary, we encode both the module's identity and its port-level roles directly into the graph's structure. This makes later queries trivial—design tools or LLM agents can retrieve functional details simply by inspecting node and edge labels.

*3) **Stage-level Grouping**:* Even after subcircuit matching and module abstraction, the circuit graph can remain highly complex: it may still contain a large number of nodes, no clear hierarchical organization, and many connections that a designer would not consider functionally important. To further organize this structure, we take inspiration from a common principle in analog circuit design, dividing the circuit into stages, and formulate a rule-based method for grouping components based on whether a continuous current can flow through them.

The core idea is that components sharing a continuous current supply path typically form a unified functional stage, operating under the same biasing conditions and serving a specific electrical role in the overall system. To implement this, we first reduce the circuit graph by removing edges that do not conduct current, such as those corresponding to capacitors or MOSFET gate connections. This results in a pruned subgraph:

$$\mathcal{G}_{\text{i}} = (\mathcal{V}_{\text{i}}, \mathcal{E}_{\text{i}}) \subseteq \mathcal{G}.$$

Within this current-conduction graph, a *conduction path* is defined as a sequence of alternating device and net nodes that connect the supply node $v_{\text{VDD}} \in \mathcal{V}_{\text{net}}$ to the ground node $v_{\text{GND}} \in \mathcal{V}_{\text{net}}$:

$$\mathcal{P}_k = (v_1, e_1, v_2, \ldots, v_n), \quad \text{with } v_1 = v_{\text{VDD}},\ v_n = v_{\text{GND}}.$$

All such paths are enumerated and collected:

$$\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_K\}.$$

Rather than treating each conduction path independently, we merge any two paths that share an internal node:

$$\mathcal{P}_i \sim \mathcal{P}_j \iff \exists v \in \mathcal{P}_i \cap \mathcal{P}_j, \quad v \notin \{v_{\text{VDD}}, v_{\text{GND}}\}.$$

This defines an equivalence relation over $\mathcal{P}$, and the union of equivalent paths forms a single stage:

$$\mathcal{S}_l = \bigcup_{\mathcal{P}_k \in [\mathcal{P}]_l} \mathcal{P}_k, \quad l = 1, \ldots, L.$$

Once the stages $\{\mathcal{S}_1, \ldots, \mathcal{S}_L\}$ are identified, we define a stage-level interconnection graph:

$$\mathcal{G}_{\text{stage}} = (\mathcal{S}, \mathcal{E}_{\text{inter}}),$$

where stages are divided and edges capture valid inter-stage signal connections. Formally:

$$(\mathcal{S}_i, \mathcal{S}_j) \in \mathcal{E}_{\text{inter}} \quad \textbf{iff} \quad \begin{cases} \exists\, v \in \mathcal{V}_{\text{net}}, \\ \exists\, u_i \in \mathcal{S}_i \cap \mathcal{V}_{\text{dev}}, \\ \exists\, u_j \in \mathcal{S}_j \cap \mathcal{V}_{\text{dev}}, \\ (u_i, v) \in \mathcal{E} \ \wedge\ (u_j, v) \in \mathcal{E}. \end{cases}$$

This ensures that stages are connected only through shared

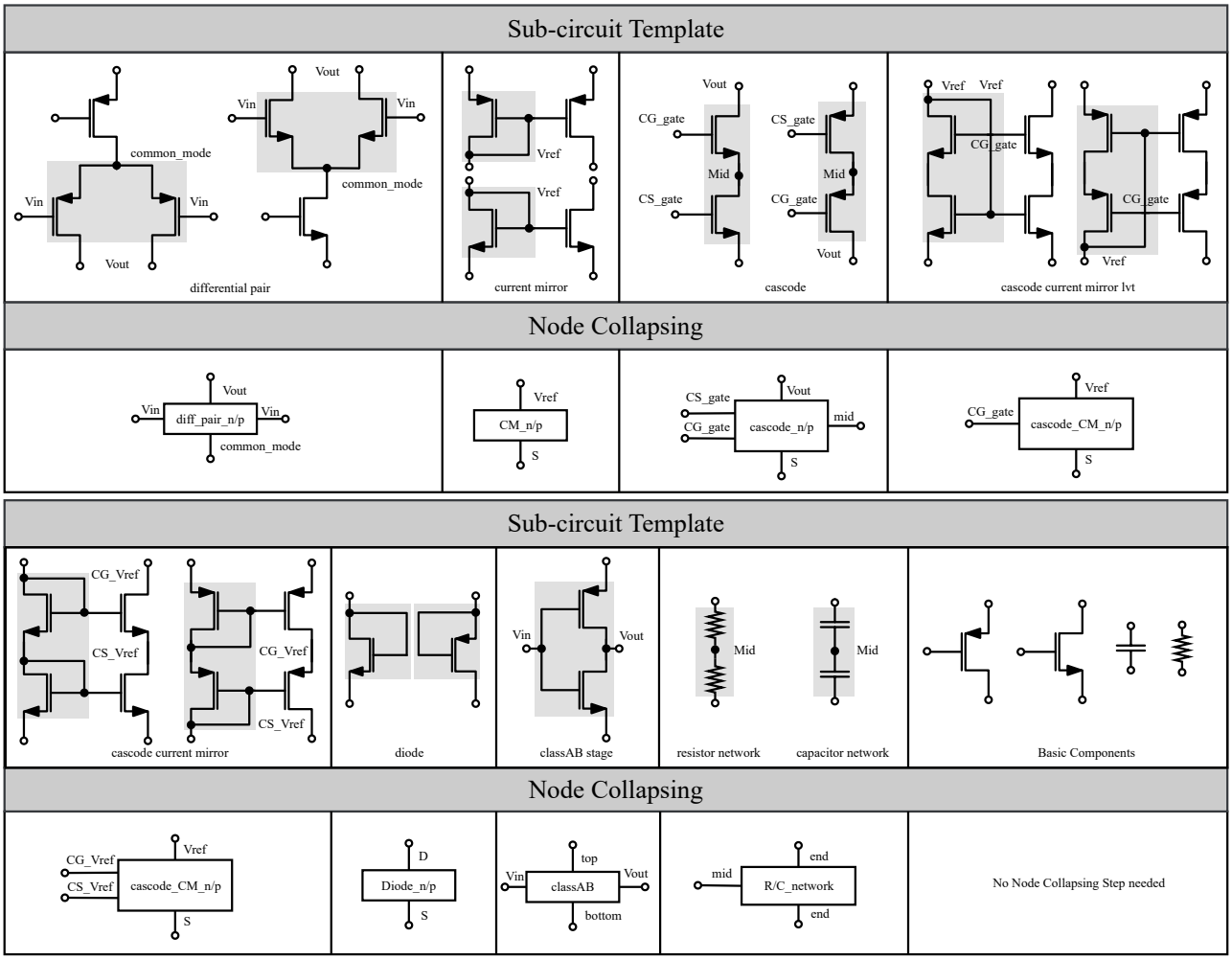


Fig. 2: The sub-circuit templates and supernode collapsing. The connection can be depicted accurately in text by embedding names into the edges and nodes.

nets, with no device-level overlap, aligning with the modular and hierarchical nature of analog circuit design. We also annotate stages containing primary input or output terminals, enabling the generation of a stage-level signal flow model. This hierarchy serves as a bridge between low-level device representation and higher-level functional architecture, providing a clear, query-friendly structure for subsequent analysis and LLM interaction.

The outcome of this step is a hierarchical graph-based container for circuit information. At the lowest level, it records individual transistors and their precise electrical connections; at the next level, these devices are grouped into recognized sub-circuits; and at higher levels, sub-circuits are organized into functional stages. This layered organization preserves complete connectivity while making it easy to query relationships—whether between specific ports, within a module, or across stages. In this way, the graph serves as an accessible and structured repository from which later stages of the framework can retrieve exactly the information they need, without losing any of the detail present in the original netlist.

C. *Circuit Understanding*

1) Iterative Circuit Understanding via LLM Agents: To achieve an accurate and robust understanding of the circuit's functional behavior, our framework employs dedicated LLM agents that iteratively analyze and refine circuit interpretations. Initially, an LLM agent processes structured circuit representations, including graph-based topology and hierarchical module information, to generate a functional analysis.

Crucially, the agent incorporates an internal confidence assessment mechanism, whereby it evaluates the certainty of its interpretations for each node, connection, or functional relationship. To ensure reliability, our model further applies a simple checklist [27] during recognition, ensuring that all device functions are successfully identified, that the functional roles hypothesized for devices within each stage are consistent with their interconnections, and that the overall

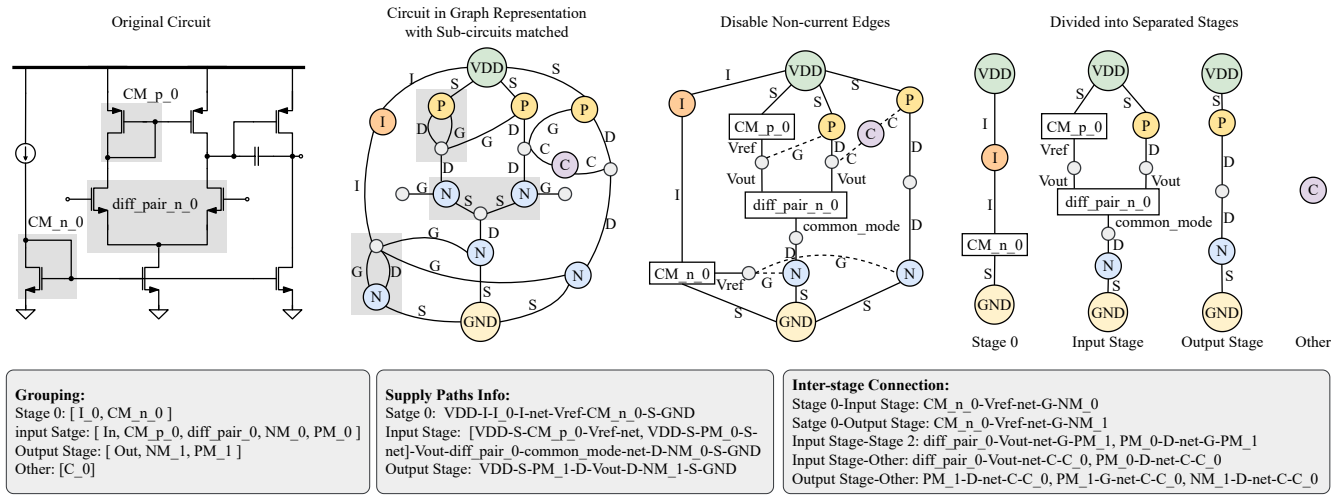


Fig. 3: Example of the grouping process of the matched circuit. The circuit topology can be then presented in text as shownn in the lower half of the figure.

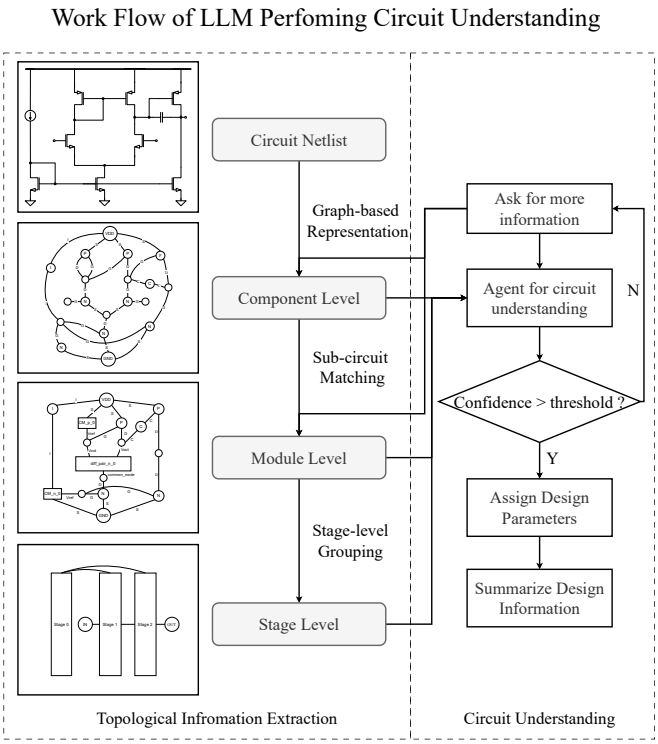


Fig. 4: LLM agent performs circuit understanding based on the hierarchical topology information extracted and stored in the previous steps.

circuit hierarchy and connectivity align with the inferred functional assignments. Only after these checks are satisfied does the agent output its result as confident. When the agent detects ambiguities or low-confidence regions, it generates targeted queries to disambiguate these uncertain elements. These queries are answered through subsequent interactions, enabling the model to iteratively refine its understanding.

The iterative process continues until the agent's confidence surpasses a predefined threshold, at which point the analysis is considered sufficiently accurate and the iteration terminates. This confidence-driven stopping criterion ensures that the model dedicates more effort to complex or uncertain circuit regions, while avoiding unnecessary computations on well-understood portions.

Throughout this workflow, a coordination agent collects and organizes the progressively refined insights into structured prompts. These comprehensive representations encode the hierarchical and functional properties of the circuit, facilitating downstream tasks such as optimization guidance and design space exploration. By integrating confidence-aware iterative analysis, the system achieves a high-fidelity understanding of complex circuit structures in an efficient and scalable manner.

2) LLM-Assisted Design Parameter Assignment: In addition to functional understanding, the framework leverages LLM agents to assign design parameters in a symmetry-aware manner. Many analog circuits contain inherently symmetric or replicated structures, such as differential pairs, current mirrors, or cascaded bias branches, where corresponding devices share identical or proportionally related sizing parameters. By automatically detecting these symmetric or functionally equivalent components from the hierarchical circuit representation, the LLM enforces consistent parameter assignments across them. This process significantly reduces the effective dimensionality of the design space, as multiple variables can be tied together under symmetry constraints rather than optimized independently. Such parameter tying mirrors common practices in manual sizing, where designers pre-define equality or ratio relationships to maintain matching characteristics and simplify the optimization process. Incorporating this capability into the automated framework not only accelerates convergence

but also ensures that the generated designs adhere to well-established analog design principles.

D. LLM-guided Optimization

Our ultimate goal is to leverage the circuit knowledge learned by the LLM to accelerate device sizing. BO is a classical and highly efficient search algorithm in device sizing, making it a natural choice as our optimization backbone. To inject domain knowledge, we leverage the LLM's circuit reasoning to intervene at two critical factors influencing BO performance: enhancing initial sampling quality through conservative space pruning, and refining trust region updates via selective guidance when stagnation occurs. This targeted integration preserves BO's efficiency while steering the search with domain-informed direction.

1) LLM-Guided Initial Sampling: The proposed LLM-guided optimization framework addresses the device sizing problem in analog circuit design, where optimal sizing parameters are strongly dependent on the specific technology library. Due to the inherent variations in device models, sizing solutions cannot be directly transferred across technologies. Consequently, the LLM cannot generate high-performance design points without prior knowledge; instead, it must rely on representative samples to infer promising regions of the design space.

To this end, the optimization process begins with an initial sampling stage. Based on the structural and functional information extracted in the previous stages, the LLM performs a loosely constrained search space pruning. The pruning is intentionally conservative to avoid excluding potentially viable designs. For example, only basic feasibility constraints are imposed, such as ensuring that the width-to-length ratio (W/L) of differential pairs exceeds 5. This eliminates parameter configurations that are clearly infeasible while retaining a broad search range.

The initial sampling is conducted primarily within the LLM-pruned subspace, ensuring that most samples are concentrated in regions more likely to yield good performance. However, to preserve global exploration capability, a controlled fraction of samples is drawn from outside the pruned region. This hybrid sampling strategy provides the optimizer with an informed and more effective initial dataset, while avoiding premature over-restriction of the search domain.

Algorithm 1 Initial Sampling

Input:
D, D_P - original and pruned design space
α - Pruned space sampling ratio
Sample $\{x_i^p\} \sim \mathcal{LHS}(P)$
Sample $\{x_i^r\} \sim \mathcal{LHS}(D \setminus P)$
$x_i \leftarrow \{x_i^p\} \cup \{x_i^r\}$ with ratio $\alpha : (1-\alpha)$
Evaluate $FoM_i \leftarrow \{FoM(x_i)\}$
return (x_i, FoM_i)

Algorithm 2 LLM-Guided TuRBO

Input: $N_{\text{init}}, N_{\text{iter}}, K,\ \alpha_{\text{inc}} > 1,\ \alpha_{\text{dec}} \in (0,1),\ r_{\min} < r_{\max}$
Randomly sample N_{init} points, fit $\mathcal{GP}_0$; set $x^\star = \arg\max \text{FoM},\ \text{FoM}^\star = \max \text{FoM},\ r_0 \in [r_{\min}, r_{\max}]$, *no_imp*$= 0$
for $t = 0$ to $N_{\text{iter}} - 1$ **do**
    Within $\mathcal{TR}_t = \{x : \|x - x^\star\|_\infty \le r_t\}$, simulate and update $\mathcal{GP}_{t+1}$
    $y_{\max} = \max \text{FoM}(X_t),\ x_{\text{new}} = \arg\max \text{FoM}(X_t)$
    If $y_{\max} > \text{FoM}^\star$**:** $(x^\star, \text{FoM}^\star) = (x_{\text{new}}, y_{\max})$, $r_{t+1} = \min(\alpha_{\text{inc}} r_t, r_{\max})$, *no_imp*=0;
    else: $r_{t+1} = \max(\alpha_{\text{dec}} r_t, r_{\min})$, *no_imp*+=1
    If *no_imp*$\ge K$: query $\mathcal{LLM}$, receive $(\tilde{x}, \tilde{r})$, set $x^\star \leftarrow \tilde{x}$, $r_{t+1} \leftarrow \text{clip}(\tilde{r}; r_{\min}, r_{\max})$, *no_imp*=0
end for
Return: $(x^\star, \text{FoM}^\star)$

2) Stagnation Triggered Trust Region Update: Following the initial sampling, the optimization proceeds using the TuRBO (Trust Region BO) algorithm [28], which adaptively explores the high-dimensional design space in parallel. While TuRBO efficiently exploits local structure, it may become trapped in sub-regions when performance improvements plateau. To address this, the LLM is incorporated as an adaptive guide for trust region adjustment.

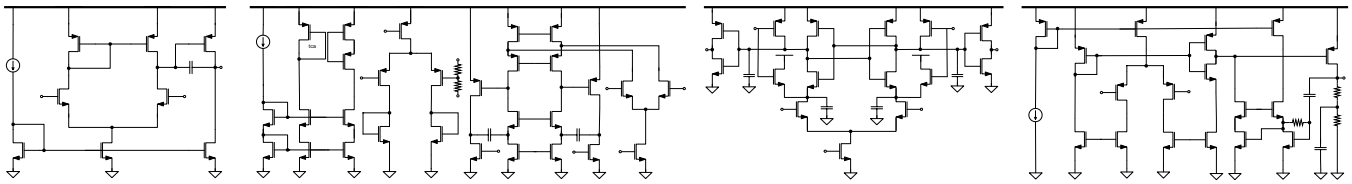
The LLM is not invoked at every iteration to reduce computational and resource overhead. Instead, its intervention is triggered by a *stagnation criterion*, specifically, when the best observed performance fails to improve over a predefined number of consecutive iterations. Upon activation, the LLM analyzes the accumulated performance data together with the topological and functional circuit information. It then generates refined constraints or directional guidance to adjust the trust region boundaries, steering the optimizer toward unexplored yet promising regions of the parameter space.

This selective intervention strategy strikes a balance between the exploitation capacity of TuRBO and the global redirection enabled by the LLM. By limiting LLM queries to moments of optimization stagnation, the framework minimizes unnecessary model calls while maintaining the ability to dynamically reorient the search toward high-performance regions.

IV. Experimental Results

A. Testcases

We select four real-world analog circuits to evaluate the design efficiency of TopoSizing: a two-stage operational transconductance amplifier (OTA), a folded-cascode operational transconductance amplifier (FCOTA), a StrongArm latch comparator (SACMP), and a low-dropout regulator (LDO). All circuits are implemented using the commercial SMIC 55nm CMOS process and are derived from practical engineering applications, ensuring their real-world relevance.



Two-stage OTA | Folded Cascode OTA | StrongArm Latch Comparator | Low Dropout Voltage Regulator

Fig. 5: Schematics of four real-world circuits.

As illustrated schematically in Fig. 5, these circuits are carefully chosen to cover a wide range of topologies, performance specifications, and design space complexities. The following sections detail the rationale behind the selection of these benchmarks and describe the corresponding experimental setups.

OTA: This circuit includes 11 design parameters: 4 transistor widths, 4 transistor lengths, 2 transistor ratios, and 1 capacitance value. The target specifications are:

$$C = \begin{cases} \text{Gain} \geq 40 \text{ dB}, & \text{GBW} \geq 50 \text{ MHz} \\ \text{Phase margin} \geq 60^\circ, & \text{Power} \leq 0.5 \text{ mW} \end{cases} \tag{3}$$

FCOTA: This circuit contains 20 design parameters: 7 transistor widths, 7 lengths, 4 ratios, and 2 capacitance values. The circuit comprises 33 components, including both core stages and biasing networks. The target specifications are :

$$C = \begin{cases} \text{Gain} \geq 100 \text{ dB}, & \text{GBW} \geq 30 \text{ MHz} \\ \text{Phase margin} \geq 60^\circ, & \text{Power} \leq 1 \text{ mW} \\ \text{PSRR} \geq 100 \text{ dB}, & \text{CMRR} \geq 100 \text{ dB} \\ \text{Noise} \leq 30 \text{ mV} \end{cases} \tag{4}$$

LDO: The LDO test case features 27 design parameters: 10 transistor widths, 10 lengths, 4 ratios, 2 resistances, and 1 capacitance. With a design space of approximately 10^{69} sampling points, it is the largest among all test cases. The target specifications are:

$$C = \begin{cases} \Delta\text{V} \leq 0.1 \text{ V}, & \text{Setup Time} \leq 15 \text{ ns} \\ \text{PSRR} \geq 60 \text{ dB}, & \text{Noise} \leq 5 \text{ uV}/\sqrt{\text{Hz}} \\ \text{Dropout} \leq 150 \text{ mV} \end{cases} \tag{5}$$

SACMP: The StrongArm latch comparator is a dynamic circuit with 14 design parameters: 6 transistor widths, 6 lengths, and 2 capacitances. It includes both symmetrical and cross-coupled structures. The target specifications are:

$$C = \begin{cases} \text{Power} \leq 40 \text{ uW}, & \text{Set Delay} \leq 4 \text{ ns} \\ \text{Reset Delay} \leq 4\text{ns}, & \text{Noise} \leq 120 \text{ uV} \end{cases} \tag{6}$$

We apply TopoSizing to all four test cases and record both the number of simulation samples and the runtime. To provide a comprehensive performance comparison, we also evaluated commercial auto-sizing tools from Cadence Virtuoso [29], as well as two academic baselines: TuRBO [28] and a reinforcement learning–based approach [4], under the same design specifications.

B. Implementation Details

We evaluate the proposed LLM-guided circuit optimization framework through a series of experiments encompassing baseline comparisons, recent LLM-aided approaches, and an ablation study. All optimization tasks are performed using TuRBO implemented in PyTorch, with circuit simulations conducted in Cadence Spectre under the SMIC 55 nm PDK. Each circuit is optimized over ten independent runs to mitigate stochastic effects, and the average results are reported. The language model adopted in all LLM-related settings is GPT-4o [30] accessed via the OpenAI API, with the generation temperature fixed at 0.5. Unless otherwise specified, no fine-tuning or task-specific pretraining is applied.

For circuit understanding, we rely on graph-based processing. Circuit netlists are first converted into graph representations, where devices and terminals correspond to nodes and electrical connections to edges. We employ `networkx` to support sub-circuit matching via subgraph isomorphism, as well as path-finding algorithms for identifying connectivity patterns between modules. This enables systematic construction of the component–module–stage hierarchy. All intermediate graph structures are stored in a standardized JSON-like format, which preserves device attributes, connectivity, and hierarchy for subsequent reasoning and optimization.

The allocation of design parameters (e.g., widths, lengths, finger numbers) to each device is supposed to be done by LLMs. Within our framework, this assignment is automatically inferred during circuit understanding and achieves complete correctness across all benchmarks. In contrast, prior LLM-based approaches that directly process raw netlists often fail to achieve correct parameter assignment, leading to ambiguous or inconsistent sizing configurations. To ensure fairness in our comparisons, we manually complete the parameter assignment step before optimization when evaluating these baselines. This guarantees that all methods begin the sizing stage from an equally valid configuration, and that the performance differences reflect only the efficiency of the LLM intervention strategy.

For non-LLM baselines, we consider a pure TuRBO implementation without language model guidance, a reinforcement learning–based optimization strategy [4], and the built-in optimizer provided by Cadence Virtuoso. For LLM-based baselines, all methods employ GPT-4o in combination with

TuRBO to ensure fairness, while differing only in the intervention strategy. Specifically, we compare our proposed adaptive intervention with a variant where the model intervenes at every optimization iteration [14], as well as the ADO-LLM approach in which two candidate points are proposed at each round [13].

To better reflect practical design requirements, experiments are conducted under two distinct scenarios: constraint satisfaction optimization, which focuses on rapidly identifying feasible solutions, and single-objective optimization, which aims to achieve progressive performance improvement over multiple iterations. In the ablation study, we assess the contributions of two key components by selectively removing either the topology analysis module or the LLM-guided enhancement to the TuRBO process, while keeping all other settings fixed.

C. Optimization Formulation

We now present the mathematical formulation of the optimization problems considered in our experiments. We first introduce the notation for the design parameters, performance metrics, performance specifications, and optimization directions:

$$\begin{aligned}\mathbf{X} &= (X_1, \ldots, X_n) \in \mathcal{D} \subseteq \mathbb{R}^n,\\ \mathbf{F}(\mathbf{X}) &= (F_1(\mathbf{X}), \ldots, F_m(\mathbf{X})),\\ \mathbf{C} &= (C_1, \ldots, C_m),\\ \boldsymbol{\varphi} &= (\varphi_1, \ldots, \varphi_m), \quad \varphi_i \in \{-1, 1\}.\end{aligned}$$

Here, $\mathbf{X}$ denotes the n-dimensional vector of design parameters, and $\mathcal{D}$ is the corresponding design space. $\mathbf{F}(\mathbf{X})$ is the m-dimensional vector of performance metrics obtained from circuit simulation, and $\mathbf{C}$ is the m-dimensional vector of performance specifications. The sign φ_i encodes the optimization direction:

$$\varphi_i = \begin{cases} 1, & \text{for metrics to be maximized,} \\ -1, & \text{for metrics to be minimized.} \end{cases}$$

1) General normalized score: For unified evaluation of metrics with different optimization directions, we define the normalized score of the i-th metric as:

$$r_i(\mathbf{X}) = \varphi_i \cdot \frac{F_i(\mathbf{X}) - C_i}{\max(|F_i(\mathbf{X})|,\, |C_i|)}, \tag{7}$$

where the numerator adjusts the sign according to the optimization direction, and the denominator normalizes the scale. A positive r_i means the i-th metric exceeds its specification, while a negative value indicates violation.

D. Multi-constraint feasibility optimization

For a pure feasibility problem, we focus on minimizing violations. The FoM is defined as:

$$\text{FoM}_{\text{feas}}(\mathbf{X}) = \sum_{i=1}^{m} \min\big(0,\, r_i(\mathbf{X})\big), \tag{8}$$

so that each constraint contributes its violation magnitude if unsatisfied, and 0 otherwise. The optimal feasible point achieves $\text{FoM}_{\text{feas}} = 0$.

1) Single-objective constrained optimization: When optimizing a specific target metric $F_t(\mathbf{X})$ while satisfying all constraints, we separate the non-target penalties from the target reward. We first quantify the penalties from all non-target metrics:

$$R_{\text{non-target}}(\mathbf{X}) = \sum_{\substack{i=1 \\ i \neq t}}^{m} \min\big(0,\, r_i(\mathbf{X})\big), \tag{9}$$

where each non-target metric contributes its violation magnitude if unsatisfied, and 0 otherwise.

The contribution of the target metric $F_t(\mathbf{X})$ depends on whether all non-target constraints are satisfied:

$$R_{\text{target}}(\mathbf{X}) = \begin{cases} r_t(\mathbf{X}), & \text{if } r_i(\mathbf{X}) \geq 0 \ \forall i \neq t, \\ \min\big(0,\, r_t(\mathbf{X})\big), & \text{otherwise.} \end{cases} \tag{10}$$

This ensures that the target metric receives a positive reward only when all other constraints are satisfied; otherwise it is also penalized if it fails to meet its own specification.

Finally, the overall figure of merit is obtained by combining the two terms:

$$\text{FoM}_{\text{single}}(\mathbf{X}) = R_{\text{non-target}}(\mathbf{X}) + R_{\text{target}}(\mathbf{X}). \tag{11}$$

E. Constraints Satisfaction

Tab. III summarizes the optimization results, averaged over ten independent runs. Compared with traditional sizing methods, our framework achieves clear improvements in both sampling efficiency and convergence speed across most test cases. Compared to other LLM-aided approaches, TopoSizing attains higher optimization efficiency while requiring substantially fewer LLM calls. Although our framework introduces additional preprocessing and circuit-understanding steps, the overall runtime remains lower, making this trade-off both practical and acceptable.

F. Single-object Optimization

We further evaluate each test case by selecting a single performance metric for single-objective optimization, in order to examine whether the proposed framework can not only accelerate optimization but also establish a correct mapping from design parameters to performance metrics. This experiment is designed to test whether LLMs, when guided through our framework, can capture such relationships more reliably and thereby enhance performance in larger-scale optimization tasks. As baselines, we compare against Bayesian Optimization (BO) with 400 samples, an LLM-based method with 400 samples, and BO with 800 samples. The experimental results confirm our expectation: except for the relatively simple 2stage case—where the LLM's interpretation is already sufficiently accurate and thus improvements are less pronounced—our framework consistently achieves comparable or better optimization outcomes with fewer simulations.

Closer inspection of the curves reveals three important observations. (1) Our framework starts from a higher FoM immediately after the initial sampling stage, indicating that

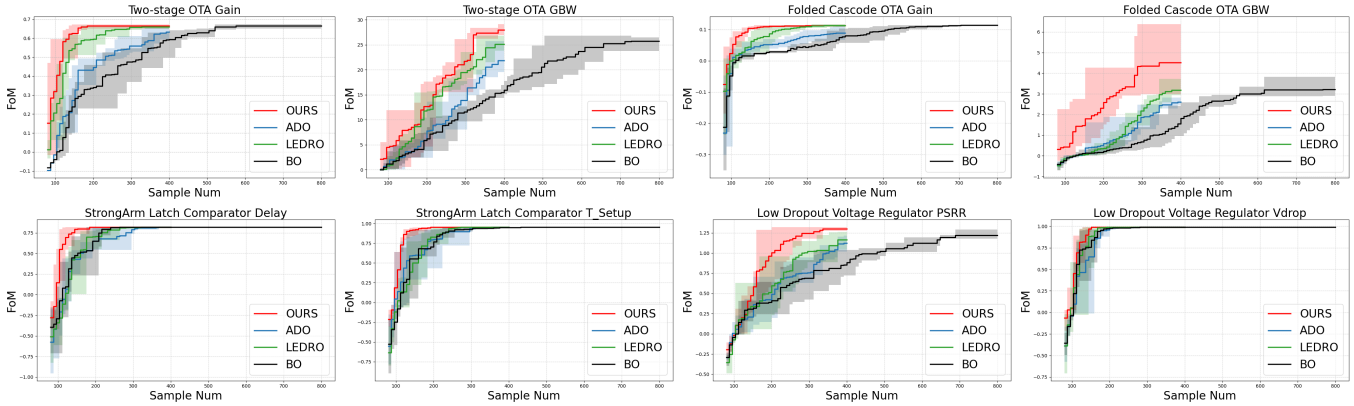


Fig. 6: FoM versus sample count in the single-objective optimization, with curves representing different methods and shaded areas showing variation across runs.

the LLM-guided initialization provides a strong advantage. (2) Long stagnation phases rarely occur, as our intervention strategy effectively invokes the LLM to adjust the search space once stagnation is detected. (3) Comparing the baselines ADO and LEDRO further illustrates the role of understanding quality: in cases where the LLM's interpretation is reliable (e.g., 2stage and FC), region-based pruning of the design space outperforms point-wise proposal strategies; in contrast, when the interpretation is less reliable (e.g., LDO), region-based modification may amplify errors and lead to performance degradation. These observations together validate both the effectiveness and the limitations of LLM-guided optimization, while highlighting the advantages of our proposed integration strategy.

G. Ablation Study

Tab. II presents the results of the ablation study, which demonstrate the effectiveness of the proposed TopoSizing framework. The analysis shows that both the topological information extraction and the iterative understanding by LLM contribute to improving circuit understanding accuracy. Furthermore, the enhanced understanding translates into higher optimization efficiency, thereby validating the overall effectiveness of our framework.

H. Case Study

Finally, we conduct a case study on the most complex FCOTA circuit to evaluate the reliability of circuit understanding in a concrete setting. As shown in Fig. 7, when directly reading the raw netlist, the LLM produces incorrect functional interpretations, failing to capture key structural relationships. In particular, for the FCOTA with a common-mode feedback (CMFB) module, the model—without any auxiliary structure—misidentifies devices around the output node(s), conflating CMFB-related elements with the main signal path. This leads to an erroneous assignment of roles in the first stage (e.g., mislabeling the load/sleeve devices and the input pair), which propagates downstream. In contrast, after applying our framework (Fig. 8), the hierarchical preprocessing and iterative verification (topological extraction, conduction-graph filtering, port-role inference, and grouping) enable the model to disentangle the CMFB loop from the differential signal path and to recover a complete and correct stage-level interpretation. This comparison highlights that our framework not only improves accuracy but also ensures robustness, eventually reaching 100% correctness through repeated verification.

TABLE I: Optimization results on four real-world circuits: Comparison With Conventional Methods

Category	Algorithm	Test Cases			
		OTA	FCOTA	SACMP	LDO
Sample #	Proposed	**18**	**32**	**18**	**22**
	TuRBO	25	66	107	37
	RL	49	86	86	60
	Virtuoso	197	142	108	400
	Eff. impr.	**1.4**×	**2.2**×	**4.8**×	**1.4**×
Runtime (s)	Proposed	**167**	**608**	**273**	**234**
	TuRBO	234	1270	1907	412
	RL	2178	4705	4407	6553
	Virtuoso	324	947	950	1141
	Speed up	**1.2**×	**1.5**×	**3.5**×	**1.4**×

TABLE II: Results of Ablation Study

Category	Algorithm	Test Cases			
		OTA	FCOTA	SACMP	LDO
Sample #	Proposed	**18**	**32**	**18**	**22**
	W/O IU*	19	52	69	39
	W/O TIE†	37	66	107	37
Accuracy of Classification	Proposed	**100%**	**100%**	**100%**	**100%**
	W/O IU	100%	87%	89%	86%
	W/O TIE	100%	48%	58%	59%

* Iterative understanding by LLM
† Topological Information Extraction

TABLE III: Optimization results on four real-world circuits: Comparison With LLM-aided Methods

Category	Algorithm	Test Cases			
		OTA	FCOTA	SACMP	LDO
Sample #	Proposed	18	**32**	**18**	**22**
	LEDRO	18	59	60	33
	ADO-LLM	24	55	72	35
	Eff. impr.	**1.0**×	**1.6**×	**2.8**×	**1.5**×
Runtime (s)	Proposed	**167**	**608**	**273**	**234**
	LEDRO	172	1328	836	407
	ADO-LLM	233	1081	972	427
	Speed up	**1.0**×	**1.8**×	**3.6**×	**1.7**×
LLM Calls #	Proposed	**1.1**	**3.3**	**2.2**	**2.1**
	LEDRO	2.3	7.4	7.5	4.1
	ADO-LLM	3.0	7.0	9.0	4.4

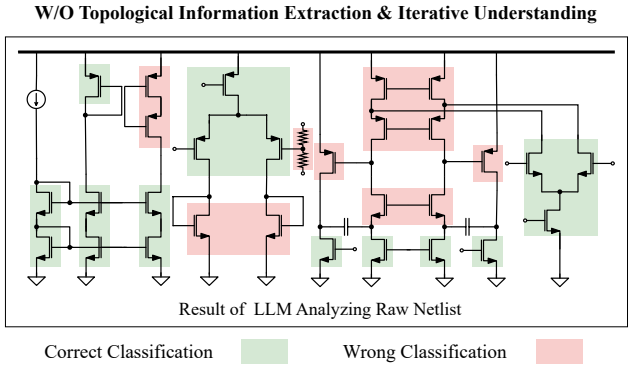


Fig. 7: LLM processing raw netlist

V. Conclusion

We propose TopoSizing, a stable and efficient topology-aware framework for analog circuit sizing. TopoSizing accelerates the optimization process through LLM integration for circuit topology comprehension and design space pruning. Experiments across four real-world test cases demonstrate its superior sampling efficiency and time efficiency compared to prior sizing tools from both industry and academia.

This fully automated and efficient design framework demonstrates significant practical value, offering a novel approach to accelerate traditional analog sizing processes through the integration of advanced AI capabilities.

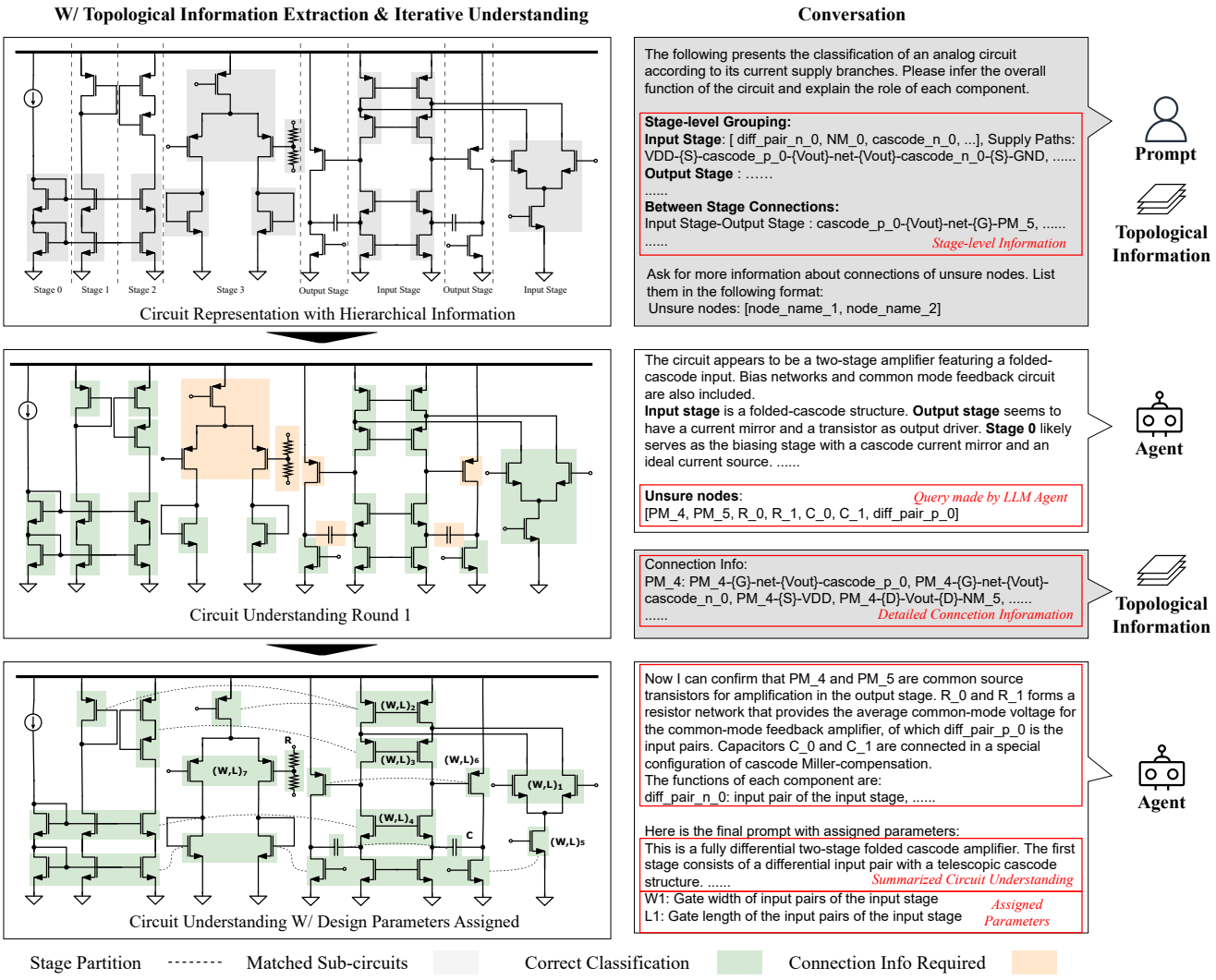


Fig. 8: The detailed steps of our iterative circuit understanding process and final output with parameter assigned.

REFERENCES

[1] R. A. Vural and T. Yildirim, "Analog circuit sizing via swarm intelligence," *AEU-International journal of electronics and communications*, vol. 66, no. 9, pp. 732–740, 2012.

[2] W. Lyu, P. Xue, F. Yang, C. Yan, Z. Hong, X. Zeng, and D. Zhou, "An efficient bayesian optimization approach for automated optimization of analog circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 6, pp. 1954–1967, 2017.

[3] K.-E. Yang, C.-Y. Tsai, H.-H. Shen, C.-F. Chiang, F.-M. Tsai, C.-A. Wang, Y. Ting, C.-S. Yeh, and C.-T. Lai, "Trust-region method with deep reinforcement learning in analog design space exploration," in *2021 58th ACM/IEEE design automation conference (DAC)*. IEEE, 2021, pp. 1225–1230.

[4] Z. Kong, X. Tang, W. Shi, Y. Du, Y. Lin, and Y. Wang, "Pvtsizing: A turbo-rl-based batch-sampling optimization framework for pvt-robust analog circuit synthesis," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.

[5] A. F. Budak, P. Bhansali, B. Liu, N. Sun, D. Z. Pan, and C. V. Kashyap, "Dnn-opt: An rl inspired optimization for analog circuit sizing using deep neural networks," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 1219–1224.

[6] A. F. Budak, D. Smart, B. Swahn, and D. Z. Pan, "Apostle: Asynchronously parallel optimization for sizing analog transistors using dnn learning," in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 70–75. [Online]. Available: https://doi.org/10.1145/3566097.3567880

[7] A. F. Budak, M. Gandara, W. Shi, D. Z. Pan, N. Sun, and B. Liu, "An efficient analog circuit sizing method based on machine learning assisted global optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 5, pp. 1209–1221, 2021.

[8] S. Kim, Z. Wang, S. Lee, Y. Oh, H. Zhu, D. Kim, and D. Z. Pan, "Ppaas: Pvt and pareto aware analog sizing via goal-conditioned reinforcement learning," *arXiv preprint arXiv:2507.17003*, 2025.

[9] H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H.-S. Lee, and S. Han, "Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[10] Z. Li and A. C. Carusone, "Design and optimization of low-dropout voltage regulator using relational graph neural network and reinforcement learning in open-source sky130 process," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 01–09.

[11] M. Liu, W. J. Turner, G. F. Kokai, B. Khailany, D. Z. Pan, and H. Ren, "Parasitic-aware analog circuit sizing with graph neural networks and bayesian optimization," in *2021 Design, automation & test in Europe conference & exhibition (DATE)*. IEEE, 2021, pp. 1372–1377.

[12] S. Lee, Z. Wang, S. Kim, T. Lee, Y. Lai, and D. Z. Pan, "Dice: Device-level integrated circuits encoder with graph contrastive pretraining," *arXiv preprint arXiv:2502.08949*, 2025.

[13] Y. Yin, Y. Wang, B. Xu, and P. Li, "Ado-llm: Analog design bayesian optimization with in-context learning of large language models," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, 2024, pp. 1–9.

[14] D. V. Kochar, H. Wang, A. P. Chandrakasan, and X. Zhang, "Ledro: Llm-enhanced design space reduction and optimization for analog circuits," in *2025 IEEE International Conference on LLM-Aided Design (ICLAD)*. IEEE, 2025, pp. 141–148.

[15] 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," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 1, 2025, pp. 379–387.

[16] J. Shen, Z. Chen, J. Zhuang, J. Huang, F. Yang, L. Shang, Z. Bi, C. Yan, D. Zhou, and X. Zeng, "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*, pp. 1–1, 2025.

[17] J. Guo, L. Du, H. Liu, M. Zhou, X. He, and S. Han, "Gpt4graph: Can large language models understand graph structured data? an empirical evaluation and benchmarking," *arXiv preprint arXiv:2305.15066*, 2023.

[18] A. Dăescu, A. Guzu, G. Nicolae, and C. Dan, "Assessing the capabilities of large language models to comprehend analog integrated circuits via netlist analysis," in *2025 3rd Cognitive Models and Artificial Intelligence Conference (AICCONF)*, 2025, pp. 1–7.

[19] Z. Tao, Y. Shi, Y. Huo, R. Ye, Z. Li, L. Huang, C. Wu, N. Bai, Z. Yu, T.-J. Lin, and L. He, "Amsnet: Netlist dataset for ams circuits," in *2024 IEEE LLM Aided Design Workshop (LAD)*, 2024, pp. 1–5.

[20] H. Xu, C. Liu, Q. Wang, W. Huang, Y. Xu, W. Chen, A. Peng, Z. Li, B. Li, L. Qi, J. Yang, Y. Du, and L. Du, "Image2net: Datasets, benchmark and hybrid framework to convert analog circuit diagrams into netlists," in *2025 International Symposium of Electronics Design Automation (ISEDA)*, 2025, pp. 807–816.

[21] W. Sun, Y. Han, B. Lan, Q. Peng, and J. Wan, "Anasizecoder: Code generator for analog integrated circuit sizing automation via large language model," in *2025 International Symposium of Electronics Design Automation (ISEDA)*, 2025, pp. 817–822.

[22] W. Chen, C. Liu, W. Huang, J. Lyu, M. Yang, Y. Du, L. Du, and J. Yang, "Analogtester: A large language model-based framework for automatic testbench generation in analog circuit design," in *2025 International Symposium of Electronics Design Automation (ISEDA)*, 2025, pp. 201–207.

[23] 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.

[24] R. Ye, C. Zhang, R. Wang, S. Xu, and Y. Zhang, "Language is all a graph needs," *arXiv preprint arXiv:2308.07134*, 2023.

[25] B. Razavi, *Design of analog CMOS integrated circuits*, 2005.

[26] R. Harjani, R. Rutenbar, and L. Carley, "Oasys: a framework for analog circuit synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 12, pp. 1247–1266, 1989.

[27] J. Cook, T. Rocktäschel, J. Foerster, D. Aumiller, and A. Wang, "Ticking all the boxes: Generated checklists improve llm evaluation and generation," 2024. [Online]. Available: https://arxiv.org/abs/2410.03608

[28] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, "Scalable global optimization via local bayesian optimization," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[29] Cadence Design Systems, Inc., "Cadence virtuoso analog design environment," 2024, accessed: 2025-08-28.

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