

# AnalogCoder-Pro: Unifying Analog Circuit Generation and Optimization via Multi-modal LLMs

Yao Lai, Souradip Poddar, Sungyoung Lee, Guojin Chen, Mengkang Hu, Bei Yu, Ping Luo, David Z. Pan

**Abstract**—Despite recent advances, analog front-end design still relies heavily on expert intuition and iterative simulations, which limits the potential for automation. We present AnalogCoder-Pro, a multimodal large language model (LLM) framework that unifies the stages of circuit topology generation and device sizing optimization. The framework features a multimodal diagnosis-and-repair feedback loop that uses simulation error messages and waveform images to autonomously correct design errors. It also builds a reusable circuit tool library by archiving successful designs as modular subcircuits, accelerating the development of complex systems. Furthermore, it enables end-to-end automation by generating circuit topologies from target specifications, extracting key parameters, and applying Bayesian optimization for device sizing. On a curated benchmark suite covering 13 circuit types, AnalogCoder-Pro successfully designed 28 circuits and consistently outperformed existing LLM-based methods in figures of merit.

**Index Terms**—Analog circuit design, large language model, topology generation, device sizing.

## I. INTRODUCTION

Large language models (LLMs) [1] have recently demonstrated strong potential for automating chip design workflows [2]–[4]. Remarkable progress has been made in digital circuit design, including RTL code generation and correction [5]–[19], script generation [19]–[21], and EDA documentation question answering [22]. While digital design tasks often benefit from standardized representations and abundant datasets, analog circuit design presents a fundamentally different set of hurdles that limit the direct transfer of such automation advances.

Analog circuit design automation, especially when applying LLMs, faces several key challenges, summarized as follows. (1) *Intricate Design Trade-offs*. Analog circuit design is characterized by tight coupling between diverse device behaviors and multidimensional performance trade-offs [41]–[45]. Design flows rely heavily on expert heuristics and extensive simulations, since even minor adjustments in topology or device sizing can drastically alter circuit functionality, leading to a combinatorial explosion in the search space. (2) *Abstraction*

The code is open-sourced at [github.com/laiyao1/AnalogCoderPro](https://github.com/laiyao1/AnalogCoderPro).

Yao Lai, Mengkang Hu, and Ping Luo are with the Department of Computer Science, The University of Hong Kong, Hong Kong SAR.

Souradip Poddar, Sungyoung Lee, and David Z. Pan are with the Department of Electrical and Computer Engineering, The University of Texas at Austin, United States.

Guojin Chen and Bei Yu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong SAR.

This work was partially done while the first author was a visiting student at UT Austin. Corresponding authors: David Z. Pan (dpan@ece.utexas.edu) and Ping Luo (pluo@cs.hku.hk).

TABLE I LLM-based Methods for Analog Design

| Work                        | Multiple Types <sup>1</sup> | Training-Free | MLLM Debug <sup>2</sup> | Circuit Gen. | Circuit Opt. | Open-Source |
|-----------------------------|-----------------------------|---------------|-------------------------|--------------|--------------|-------------|
| LADAC [23]                  | •                           | •             |                         |              | •            |             |
| ADO-LLM [24]                | •                           | •             |                         |              | •            |             |
| LaMAGIC [25], [26]          |                             |               |                         | •            | •            |             |
| AnalogCoder [27]            | •                           | •             | ◦                       | •            |              | •           |
| SPICEPilot [28]             | •                           |               | ◦                       | •            |              |             |
| LEDRO [29]                  |                             | •             |                         |              | •            |             |
| Aritsan [30]                |                             |               |                         | •            | •            |             |
| AmpAgent [31]               |                             | •             |                         |              | •            |             |
| Atelier [32]                |                             | •             |                         | •            | •            |             |
| AnalogXpert [33]            |                             | •             | ◦                       |              |              |             |
| Malasa-Chai [34]            | •                           |               |                         | •            |              | •           |
| AnalogGenie/Lite [35], [36] | •                           |               |                         | •            | •            | •           |
| AnalogFed [37]              | •                           |               |                         | •            |              |             |
| AutoCircuit-RL [38]         |                             |               |                         | •            | •            |             |
| SPICEAssistant [39]         |                             | •             | ◦                       | •            | •            |             |
| MenTeR [40]                 | •                           | •             | ◦                       | •            | •            |             |
| <b>AnalogCoder-Pro</b>      | •                           | •             | •                       | •            | •            | •           |

<sup>1</sup> Whether the work supports multiple circuit type designs.

<sup>2</sup> • - Full multi-modal debugging capability by signal images (e.g., waveform, frequency response), ◦ - Text-only debugging.

*gap*. Digital design languages such as Verilog [46] provide high-level functional abstractions, enabling concise descriptions (e.g., a digital adder can be specified in a single line of code). In contrast, analog design inherently operates at the device level, where even basic functions such as addition require explicitly wiring together multiple transistors and passive components. This abstraction gap limits the transfer of automation advances from digital to analog design. (3) *Data scarcity*. Public code repositories contain very little hardware description language content—Verilog, for instance, accounts for less than 0.1% of GitHub projects [47]. SPICE [48], the predominant analog design language, has an even smaller footprint, making it particularly challenging for LLMs to acquire analog design rules effectively. (4) *Underuse of multimodal information*. Simulation outputs such as waveforms and frequency responses contain critical information for analog circuit analysis. Yet, these non-textual modalities are rarely integrated into automated processing pipelines. This underutilization limits the ability of LLMs to diagnose issues and refine designs effectively. (5) *Lack of unified topology–sizing optimization*. Topology generation and device sizing are typically conducted as separate, sequential stages, which can lead to suboptimal results. When inherent topological constraints cannot be addressed through parameter tuning, costly redesign cycles often follow.

Conventional analog automation tools provide valuable solutions but still face limitations [49]. Modern device sizing algorithms [50]–[60] improve sample efficiency yet are mostly

restricted to predefined architectures. Topology exploration methods face high simulation costs [61], [62], depend on expert-crafted equations [63]–[65], offer limited diversity [66], and carry the risk of generating invalid structures [67]. In addition, automated synthesis approaches such as [68] still demand substantial setup effort, limiting the applicability to predefined libraries.

Recent LLM-based approaches for analog circuit design, partly shown in TABLE I, have emerged in three main directions, including topology generation via code or netlist synthesis [27], [28], [33], [34], [40], behavioral-level synthesis [30], [32], circuit sizing [23], [24], [31], [69], [70], layout automation [69], [71], test-bench generation [72], and question–answering assistants [73]–[76]. While promising, these methods lack comprehensive solutions that integrate multimodal feedback and unified topology–sizing optimization. For example, AnalogCoder [27] and AnalogXpert [33] have not yet incorporated waveform images for circuit analysis. AnalogGenie [35] supports end-to-end topology generation through pre-training and fine-tuning, yet still depends on manual data collection and per-class fine-tuning, which restricts scalability.

To overcome these limitations, we propose AnalogCoder-Pro, a training-free and end-to-end multimodal LLM framework that jointly performs topology generation and device sizing via Python code generation. To the best of our knowledge, AnalogCoder-Pro is the first framework in this direction. Given a natural-language description of the target circuit’s functionality, structure, and performance goals, the framework autonomously generates, verifies, and optimizes analog circuits. For instance, when asked to “Design a multi-stage opamp that optimizes the FoM (GBW/Power),” it can produce and iteratively refine a functional design. The key innovations of AnalogCoder-Pro include domain-specific design prompts, a feedback-enhanced flow that leverages multimodal inputs such as waveforms, a dedicated circuit tool library that enables flexible reuse of fundamental analog blocks, and LLM-assisted parameter extraction for circuit sizing. Together, these components allow the framework to deliver high-quality, verified analog front-end designs with minimal human intervention.

The main contributions of this work are:

- Introduction of AnalogCoder-Pro, the first training-free multimodal LLM framework for end-to-end analog circuit design, capable of transforming natural-language design objectives into optimized netlists.
- Development of a feedback-enhanced multimodal design flow that leverages waveforms, simulation logs, and code diagnostics to iteratively refine designs and correct functional errors.
- Creation of a circuit tool library that enables flexible reuse and composition of fundamental analog blocks for complex circuit design.
- Proposal of an integrated parameter optimization framework that combines LLM-generated initial topologies with automated refinement for effective circuit sizing.
- Experimental validation showing that AnalogCoder-Pro outperforms existing LLM-based methods in both design

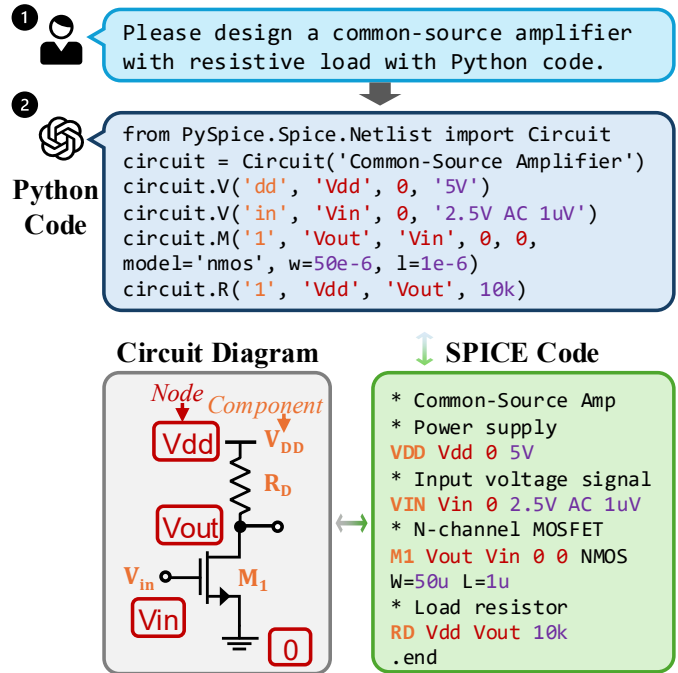


Fig. 1 Code-based Analog Circuit Representation. Given a design task, the LLM generates Python code with the PySpice library. The circuit can be equivalently represented as (1) a circuit diagram, (2) SPICE code, or (3) PySpice-based Python.

success rate and circuit performance across diverse analog circuit types.

## II. PRELIMINARIES

### A. Analog Circuit Design Challenges and Representation

**Design Challenges.** Analog circuit design is one of the most intricate stages in the integrated circuit flow, involving both front-end and back-end tasks. In the front-end, designers must synthesize suitable topologies and optimize device parameters to meet stringent, often conflicting performance criteria such as gain, bandwidth, and power consumption. Achieving a high figure of merit (FoM) requires navigating large, high-dimensional design spaces and managing complex trade-offs, which are further complicated by the strong coupling between topology and sizing decisions, where an optimal choice in one aspect may be suboptimal in another. The back-end translates these logical designs into manufacturable layouts [71], [77]–[81], ensuring process compliance.

Recent advances in artificial intelligence have brought new tools to analog design, including reinforcement learning [50], [52], [82], [83], graph-based generative models [84], rule-based topology synthesis via graph grammars [85], diffusion-based models [86], and language model-based approaches [23], [24], [27]–[29], [31], [33], [34], [58]. Nevertheless, many of these techniques tackle topology generation and parameter optimization in isolation, which can result in incompatibilities that hinder overall performance and necessitate iterative redesigns.

To mitigate these issues, recent research has explored uni-

fied frameworks that jointly consider topology and sizing. AnalogGenie [35], [36] first pre-trains on a manually collected set of circuit topologies, then fine-tunes the model specifically on high-performance topologies. The method in [68] automatically ranks candidate topologies according to user requirements, then performs sizing and verification on each, but remains limited to a predefined set of topologies. Artisan [30] focuses on behavioral-level synthesis for op-amps design, omitting device-level refinement. Thus, despite these advances, a truly scalable, general-purpose methodology for end-to-end, performance-driven analog design automation remains elusive.

**Circuit Representation.** SPICE [48] is the industry-standard netlist format for circuit-level modeling and simulation, in which circuits are described as interconnected elements (e.g., resistors, voltage sources, MOSFETs) with associated parameters, as illustrated by the amplifier example in Fig. 1. In our framework, we integrate PySpice [87], a Python interface to SPICE simulators (e.g., Ngspice) that enables circuit construction, simulation control, and results parsing within a unified scripting environment. This approach leverages Python’s programmability and ecosystem for efficient circuit generation, simulation, and analysis, and aligns with the strong capability of LLMs in Python code generation [88]. Consequently, PySpice serves as a natural entry point for LLM-based circuit synthesis, replacing manual netlist creation.

### B. Multimodal Large Language Models

Multimodal large language models (MLLMs) extend traditional LLMs by jointly processing visual and textual inputs. An MLLM typically includes a vision encoder and a text encoder, which produce feature tokens from images  $I$  and text  $T$ :

$$z_{\text{vision}} = E_{\text{vision}}(I), \quad z_{\text{text}} = E_{\text{text}}(T), \quad y = \text{LM}([z_{\text{vision}}, z_{\text{text}}]),$$

where  $y$  is the output sequence.

In EDA, MLLMs enable richer interpretation of circuit diagrams and waveform plots, outperforming text-only models in pattern recognition tasks. For example, Fig. 2 shows that a noisy triangular waveform is misclassified by an LLM (text input only) as a damped sinusoid, whereas an MLLM correctly identifies it from the waveform image. This visual reasoning capability is crucial for reliable waveform identification, thereby improving circuit debugging and validation.

## III. FRAMEWORK AND METHODS

### A. Framework Overview

AnalogCoder-Pro is a multimodal LLM-based framework that interprets natural-language task descriptions to automatically generate Python code representing functional, optimized analog circuits. As shown in Fig. 3, it consists of multiple modules, including prompt engineering, a multimodal feedback-enhanced flow, and a circuit tool library, followed by device sizing. Prompt engineering enhances the design reasoning through strategic, problem-solving prompts. The multimodal feedback-enhanced flow incorporates multiple checks, including waveform validation, to feed error messages back to the LLM, enabling it to correct failed designs. The circuit tool library serves as a modular repository that organizes designed

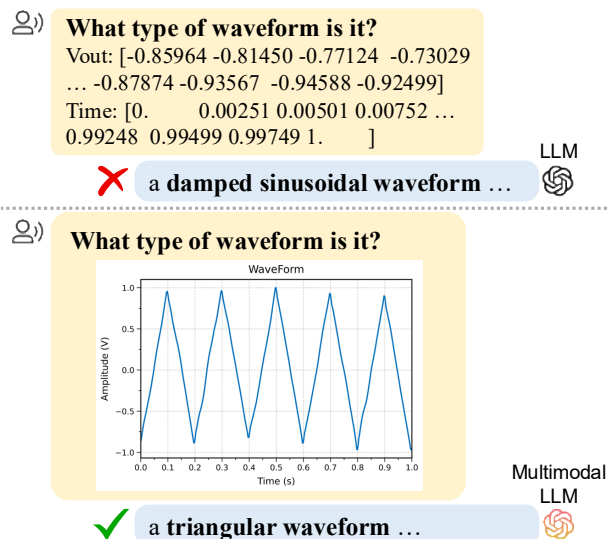


Fig. 2 LLMs vs. MLLMs in Waveform Identification. LLMs provided with textual data (top) misidentify a noisy triangular waveform, whereas MLLMs, given the corresponding waveform image (bottom), correctly identify its type.

circuits into reusable components, enabling efficient retrieval and integration into more complex designs. Following topology generation, it conducts device sizing, including parameter extraction and optimization, to finalize the circuit design.

### B. Prompt Design

To leverage the code generation capabilities of LLMs for circuit design, we adopt a systematic prompt design strategy, comprising: (1) *Programming Language Selection*, (2) *In-Context Learning* [89], and (3) *Chain-of-Thought reasoning* [90]. Given the superior performance of LLMs in Python code generation [88], owing to the prevalence of Python in model pre-training, we instruct the models to produce executable Python code compatible with the PySpice library, thereby overcoming the lack of SPICE code in common training datasets [47]. For in-context learning, we provide a single detailed example of a two-stage amplifier, enabling the LLM to generalize and standardize its output across diverse design tasks. To further enhance design quality, we employ Chain-of-Thought prompting [90], which guides the LLM to generate a step-by-step design plan before producing the corresponding code.

### C. Multimodal Feedback-enhanced Flow

To ensure the correctness of analog circuit designs, the verification in feedback flow is divided into five stages as shown in Fig. 4: (1) *Requirement Check*, which ensures basic design elements such as the input voltage ( $V_{in}$ ) and output voltage ( $V_{out}$ ) are present, and validates the inclusion of essential components; (2) *Simulation and Operating Point Check*, where the circuit is simulated to detect issues like floating nodes and verify MOSFET operating conditions (e.g.,  $V_{gs} > V_{th}$ ); (3) *DC Sweep Check*, which involves varying  $V_{in}$  to observe changes in  $V_{out}$ , ensuring signal path integrity and identifying optimal bias points; (4) *Function Check*, where simulations validate functionalities such as gain and periodicity; and (5)

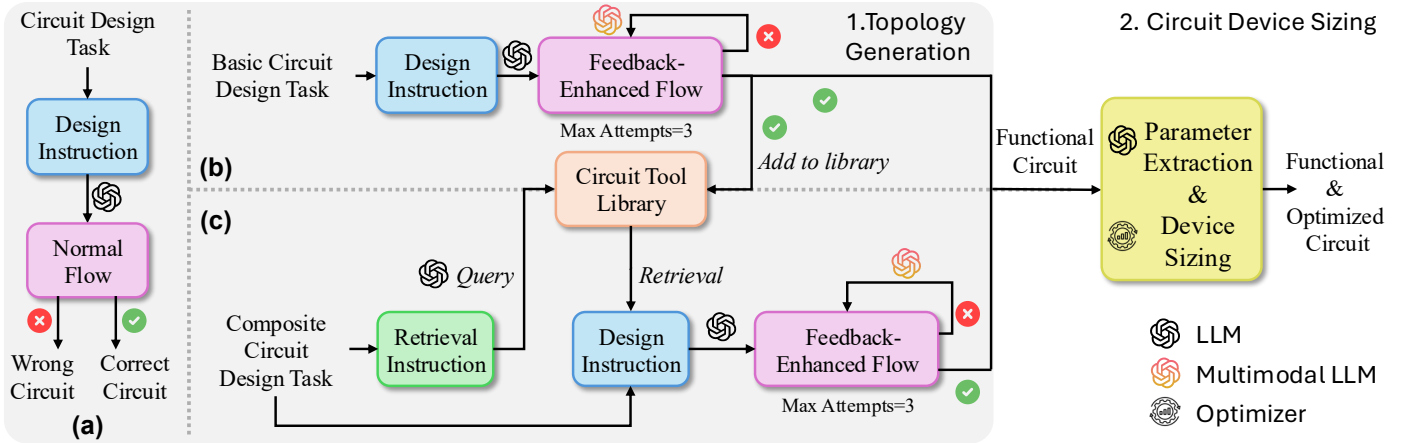


Fig. 3 Framework Overview. AnalogCoder-Pro consists of two main stages: (1) topology generation and (2) circuit device sizing. (a) Standard method. A direct design flow often results in failed circuits. (b) AnalogCoder-Pro for basic circuits. design prompts are processed through a multimodal feedback-enhanced flow, where LLMs iteratively correct errors; successful circuits are stored in the circuit tool library. (c) AnalogCoder-Pro for composite circuits. Relevant subcircuits are retrieved from the library and incorporated into the design prompt to facilitate composite circuit generation. After topology generation, the device sizing is conducted, which gives the final functional and optimized circuit.

TABLE II Waveform Types for Different Circuit Types

| Circuit Type    | Abbr. | Transient | AC | FFT | DC Sweep | DC Transfer |
|-----------------|-------|-----------|----|-----|----------|-------------|
| Mixer           | Mix   | •         |    | •   |          |             |
| Comparator      | Comp  |           |    |     | •        |             |
| Filter          | Filt  |           | •  |     |          |             |
| Oscillator      | Osc   | •         |    | •   |          |             |
| Integrator      | Int   | •         |    |     |          |             |
| Differentiator  | Diff  | •         |    |     |          |             |
| Schmitt trigger | Sch   | •         |    |     |          |             |
| VCO             | Vco   | •         |    | •   |          |             |

*Waveform Check:* simulation data are plotted as waveforms (e.g., transient, AC sweep) using the Matplotlib package, analogous to an oscilloscope, to visually verify the output signal. The used waveform types for different circuit types are shown in TABLE II.

If errors are detected during these stages, the framework generates three types of feedback signals: (1) template-based diagnostic messages from our checking scripts (e.g., “Missing  $V_{in}$  node”, “current  $I_D$  for Mosfet is 0.”); (2) simulator logs from both stdout and stderr, covering numerical results and execution warnings; and (3) waveform analysis capturing the circuit’s dynamic behavior. Regarding the third signal, since our main LLM operates entirely in the textual domain, waveform images are converted into concise textual descriptions by a multimodal LLM, a process exemplified in Section III-E. The main LLM then leverages this combined feedback to regenerate the circuit code, correcting potential errors. For instance, if a band-stop filter fails to exhibit the expected band-stop magnitude response, the multimodal model summarizes this deviation in text, guiding the LLM to adjust component values. The framework allows up to three revision cycles to balance token usage with repair effectiveness.

#### D. Circuit Tool Library

As analog circuit design tasks grow more complex, it becomes increasingly challenging for LLMs to generate correct circuits

from scratch. To address this, we modularize basic circuits into subcircuits, facilitating their integration into composite designs. Inspired by tool-based LLM studies [91], [92], we developed a circuit tool library to store and reuse correctly designed subcircuits. As illustrated in Fig. 5, our approach comprises two key processes: (1) *Library Construction from Basic Designs*. After the LLM completes a basic circuit design, the corresponding code is stored in the library, where subcircuit descriptions serve as queryable keys, while the netlist code and usage instructions serve as values. (2) *Library-Assisted Composite Design*. When presented with a high-level design goal for a composite circuit, the framework leverages the library in a two-step process. First, in the tool selection step, the LLM acts as a system planner. It analyzes the functional requirements of the composite task description. Through its reasoning capabilities, the LLM decomposes this high-level goal into constituent functional blocks. It then maps these required blocks to the available tools in the library by matching the task’s functional needs against the queryable keys (subcircuit descriptions). For instance, when tasked with designing an “op-amp integrator,” the LLM deduces the necessity of an “operational amplifier” subcircuit and selects the corresponding tool from the library. Second, in the guided generation step, the retrieved subcircuit’s code and usage instructions are used to formulate an augmented prompt. This prompt guides the LLM to instantiate the subcircuit and connect it with other components to form the final design. For the op-amp integrator example, the prompt would contain the task description alongside the specific op-amp subcircuit call, ensuring the design is built around this pre-verified block.

#### E. Topology Generation Example

Below is a complete example conversation for circuit topology generation in AnalogCoder-Pro. The blue text in the User Prompt question part represents user-specified design require-

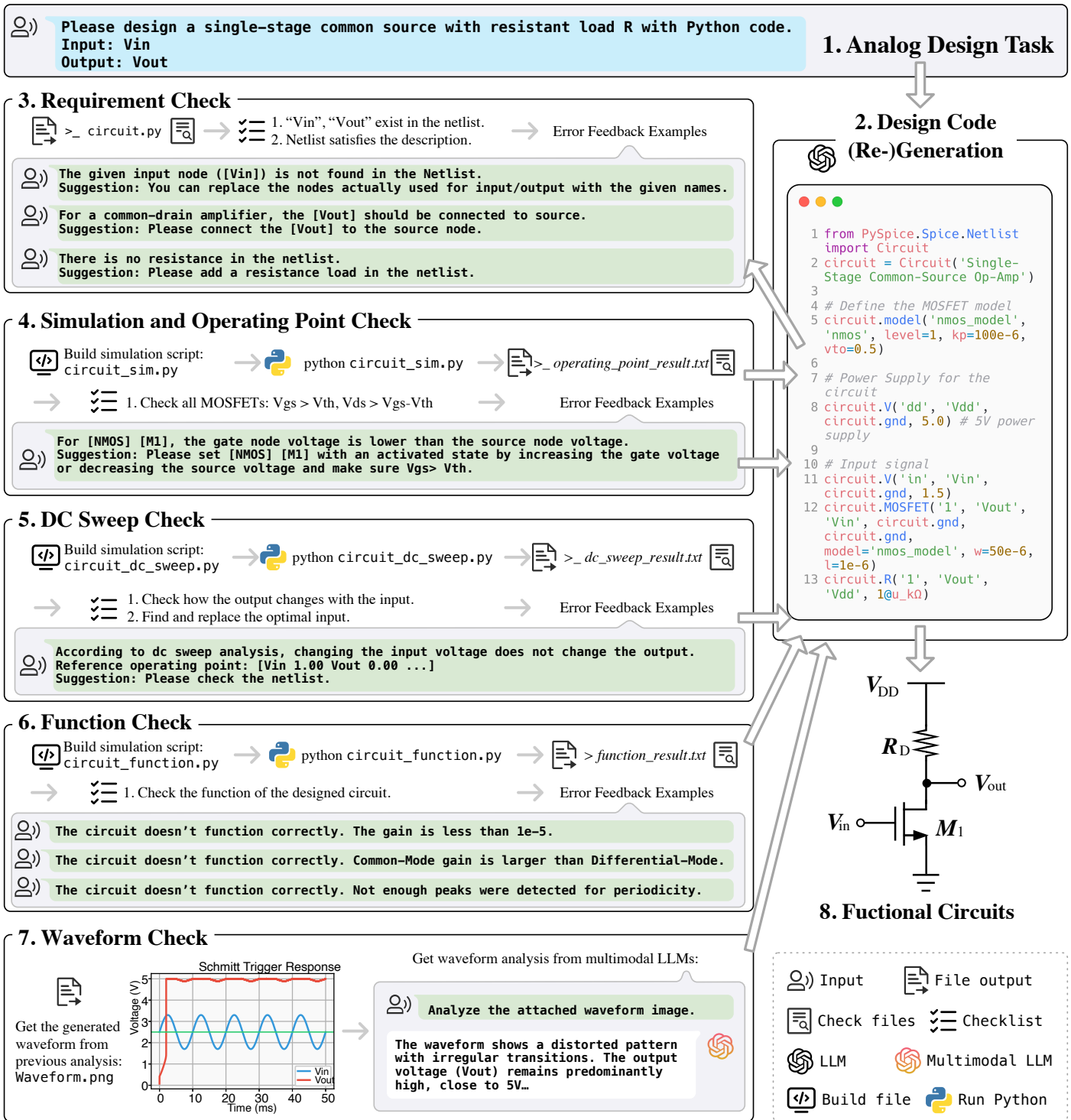


Fig. 4 Multimodal Feedback-Enhanced Design Flow. A feedback-enhanced flow in which the LLM integrates textual error messages and waveform anomalies to iteratively refine designs without human intervention. Error messages are generated from predefined templates, ensuring applicability across diverse circuit types. Waveform images are automatically produced from simulation data using the matplotlib package. For illustration, the waveform example is derived from a Schmitt trigger circuit.

ments, while the rest is fixed prompt text. In this example, the LLM is tasked with designing an RC-shift oscillator, with the op-amp subcircuit from the circuit tool library provided for reuse. The simulation process will generate the transient

waveform image, and the MLLM feedback-enhanced flow is introduced for circuit diagnosis and repair. We first give the initial round of conversation.

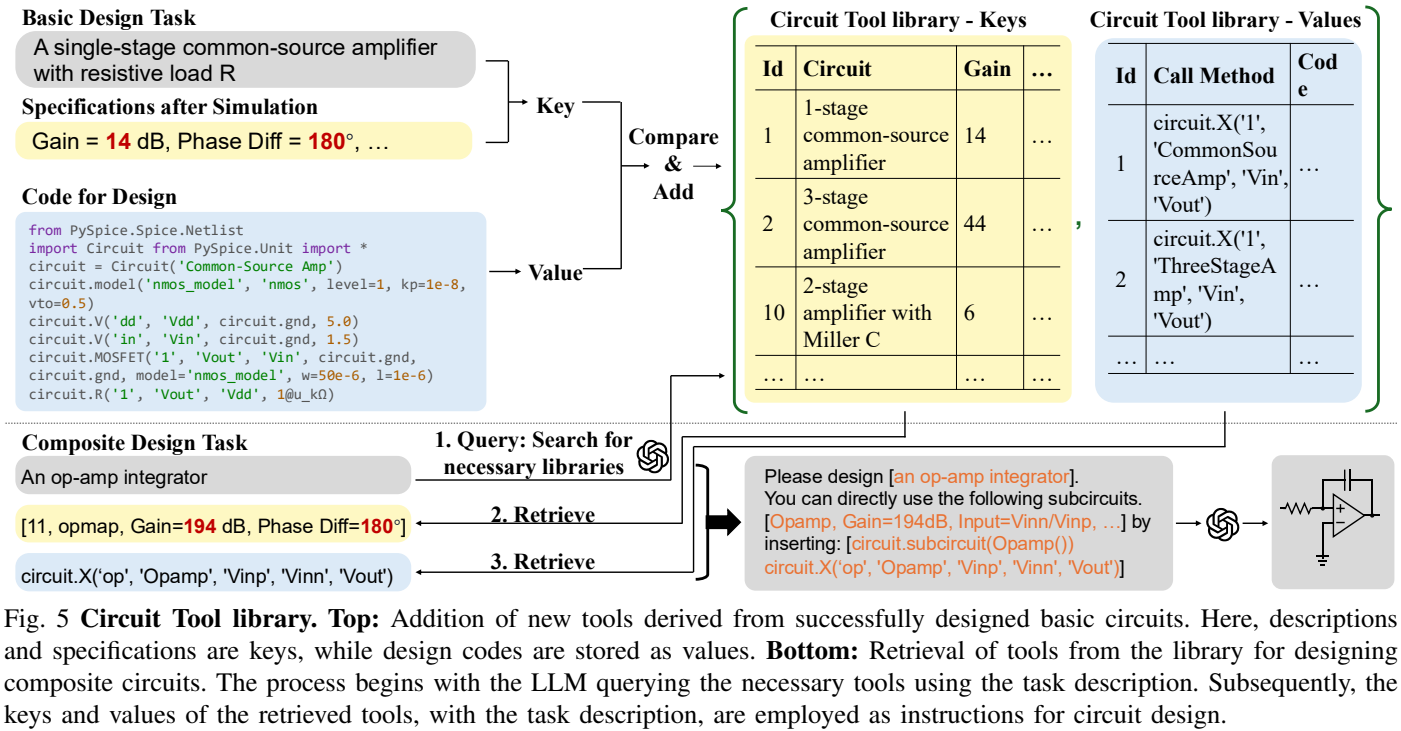


Fig. 5 **Circuit Tool library.** **Top:** Addition of new tools derived from successfully designed basic circuits. Here, descriptions and specifications are keys, while design codes are stored as values. **Bottom:** Retrieval of tools from the library for designing composite circuits. The process begins with the LLM querying the necessary tools using the task description. Subsequently, the keys and values of the retrieved tools, with the task description, are employed as instructions for circuit design.

**User Prompt (Round 1; LLM Input)**

You aim to design a topology for a given circuit described in the text. Please ensure your designed circuit topology works properly and achieves the design requirements. To make the task easier, you can directly use the opamp subcircuits by inserting:

```

from opamp import *
# Declare the operational amplifier subcircuit
circuit.subcircuit(Opamp())
# Create a subcircuit instance
# Parameter order: instance name, subcircuit name,
# the non-inverting input, the inverting input,
# the output
circuit.X('1', 'Opamp', 'Vinp', 'Vinn', 'Vout')
    
```

The DC bias voltage for both input terminals (Vinn and Vinp) is 2.5V. For AC coupling/grounding purposes, these terminals should be referenced to the 2.5V power supply rather than ground.

Here is an example:  
 ...*(An example code for an opamp circuit with resistive load)*

As you have seen, the output of your designed topology should be in a complete Python code, describing the topology of integrated analog circuits according to the design plan. Please make sure your Python code is compatible with PySpice. Please give the runnable code without any placeholders.

There are some tips you should remember all the time:

- For the MOSFET definition `circuit.MOSFET(name, drain, gate, source, bulk, model, w=w1, l=l1)`, be careful about the parameter sequence.
- You should connect the bulk of a MOSFET to its source.
- Use nominal transistor sizing.
- ...

Please first give a detailed design plan and then write the code.

**Question**

Design an RC phase-shift oscillator.  
 Input node name: -.  
 Output node name: **Vout**.

**Answer**

**AI Response (Round 1; LLM Output)**

An RC phase-shift oscillator uses an amplifier with a feedback network consisting of three RC stages. Each RC stage provides approximately 60° phase shift, for a total of 180° through the feedback network. Combined with the 180° phase shift from the inverting amplifier, this creates a 360° phase shift around the loop, satisfying the phase criterion for oscillation.

**Design Requirements:**

- Create an oscillator circuit using an operational amplifier.
- Implement three RC stages for phase shifting.
- Set appropriate gain to ensure oscillation.
- ...

**Circuit Components:**

- Operational amplifier in inverting configuration.
- Three RC stages for phase shifting.
- Feedback resistors to set appropriate gain.

**Design Details:**

- For an RC phase-shift oscillator to work, the amplifier gain must be at least 29 (approximately 29.3) to overcome losses in the RC network.
- Three identical RC stages will be used, with  $R = 10\text{ k}\Omega$  and  $C = 10\text{ nF}$ .
- The oscillation frequency will be approximately  $f = \frac{1}{2\pi RC\sqrt{6}} \approx 650\text{ Hz}$ .
- The feedback resistor  $R_f$  will be set to provide sufficient gain ( $> 29$ ).
- The input resistor  $R_{in}$  will be chosen to achieve this gain.

Let's implement this design:

```

# ... (Imports and Model Definitions omitted) ...

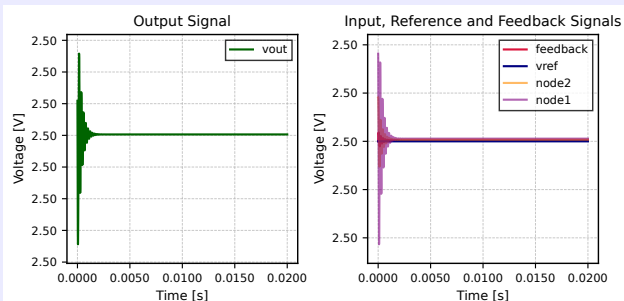
# RC phase-shift network (three stages)
circuit.R('1', 'Vout', 'node1', 10@u_kOhm)
circuit.C('1', 'node1', 'Vref', 10@u_nF)
# ... (Stages 2 and 3 omitted, identical structure) ...

# Feedback resistor (The problematic part)
circuit.R('f', 'feedback', 'Vout', 300@u_kOhm)
# Input resistor
circuit.R('in', 'feedback', 'Vref', 10@u_kOhm)
    
```

During functional testing, the oscillator failed to maintain stable oscillation—the amplitude gradually decreased to zero. The waveform, generated with Matplotlib, was submitted to a multimodal LLM for further analysis. We give an example as follows. The parts highlighted in orange will be updated according to the circuit type.

### User Prompt (Multimodal LLM Input)

Analyze the attached waveform output image from an RC phase-shift oscillator circuit which is designed by one LLM. **Vout should exhibit periodic oscillations.** The actual output waveform differs. The testbench initializes the op-amp non-inverting input (Vinp) to 2.51V and the inverting input (Vinn) to 2.5V. Please provide a detailed analysis of the waveform characteristics within one paragraph ...



### AI Response (Multimodal LLM Output)

The waveform shows a damped oscillation that quickly settles to a steady DC level rather than exhibiting the expected sustained periodic oscillations for an RC phase-shift oscillator. The output signal (vout) initially displays several rapidly decaying oscillations with a peak amplitude deviation of approximately  $\pm 0.01V$  around the 2.5V level, occurring within the first 0.001 seconds of simulation time.

The waveform analysis from the MLLM and the error messages obtained during the check process are jointly fed back to the LLM, enabling it to revise the design and regenerate a correct circuit implementation.

### User Prompt (Round 2; LLM Input)

I am encountering an error when running the PySpice code. Below is the error message:  
 Average oscillation period: 0.000002 s  
 Maximum amplitude: 0.000000 V  
 The oscillation amplitude is too small.

#### Waveform Analysis:

The waveform shows a damped oscillation that quickly settles to a steady DC level rather than exhibiting the expected sustained periodic oscillations for an RC phase-shift oscillator. ... (the output of MLLM waveform analysis)

Identify any structural errors in this circuit's topology (e.g., missing components, incorrect connections) and rewrite the complete corrected code from scratch, with each modification marked and explained in comments.

### AI Response (Round 2; LLM Output)

#### Corrected RC Phase-Shift Oscillator Design

The main issues with the previous design were:

- Insufficient gain to sustain oscillations
- Improper RC network configuration
- Lack of initial condition to start oscillations

Let me redesign the circuit with proper topology and gain:

```
# ... (Previous setup code remains the same) ...

# CORRECTION: Significantly increased gain (100)
circuit.R('F', 'feedback', 'Vout', 1000@u_kOhm)

# CORRECTION: Add initial condition
# to kickstart oscillation
circuit.C('kickstart', 'Vout', 'Vref', 1@u_nF,
         initial_condition=0.1@u_V)

simulator = circuit.simulator()
```

#### Key corrections:

- **Increased gain:** The gain of the amplifier is now much higher ( $R_f/R_{in} = 1000k/10k = 100$ ), well above the minimum required gain of 29 for an RC phase-shift oscillator.
- **Proper RC network:** The RC network is configured correctly with each capacitor connected to the reference voltage.
- **Initial condition:** Added a capacitor with an initial condition to kickstart the oscillation.
- **Feedback path:** Ensured proper feedback from the last RC stage to the inverting input of the op-amp.

This circuit should now produce sustained oscillations at approximately 650 Hz.

Based on the analysis of the waveforms, the oscillator was revised primarily by adjusting the feedback resistor value to increase the loop gain. This modification enables the circuit to sustain stable oscillations. The post-simulation waveform of the revised oscillator is shown in Fig. 6.

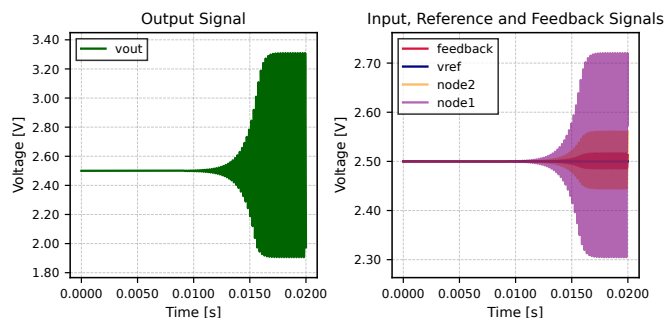


Fig. 6 Waveform of a functional RC Phase-Shift Oscillator

#### F. LLM-assisted Circuit Device Sizing

Upon completion of circuit topology generation, the resulting code contains nominal parameters (e.g., transistor sizing, bias voltages, capacitances, resistances) that ensure functionality but may be suboptimal in performance. We address this by instructing the LLM to: (1) analyze the generated netlist, (2) extract all tunable parameters with appropriate search ranges, and (3) reformat the circuit into a parameterized template for use with optimization frameworks (e.g., Optuna). We give an example instruction and response as follows. In this case, the LLM extracts the resistance value and the transistor width as tunable parameters, producing a parameterized circuit implementation that an optimization framework can invoke.

### User Prompt (LLM Input)

I need to convert my PySpice circuit netlist into a parameterized form for automated optimization. Please transform the following original circuit description into a parameter-based implementation with appropriate search ranges.

For reference, here is an example of what an original circuit might look like: ...

Here is my actual circuit code:

```

circuit = Circuit('Single-Stage Amplifier')
# MOSFET model
circuit.model('nmos_model', 'nmos', **nmos_params)
circuit.model('pmos_model', 'pmos', **pmos_params)
# Power Supply
circuit.V('dd', 'Vdd', circuit.gnd, 1.2)
circuit.MOSFET('1', 'Vout', 'Vin', circuit.gnd,
              circuit.gnd, model='nmos_model', w=5e-6, l=0.045e-6)
circuit.R('1', 'Vout', 'Vdd', 10@u_kOhm)
    
```

### Required Output

- 1) A circuit creation function that uses a `params` dictionary.
- 2) A parameter search range dictionary (for Optuna or similar optimization tools).
- 3) Reasonable parameter constraints with tailored search ranges:
  - Transistor width ( $W$ ) should be within  $1\text{--}500\times$  the corresponding length ( $L$ ).
  - All voltage sources (except  $V_{dd}$  and  $V_{in}/V_{inp}/V_{inn}$ ):  $0.8\text{--}1.2\times$  original value, not exceeding  $1.2\text{ V}$
  - Resistors:  $0.5\text{--}2\times$  original value
  - ...

### AI Response (LLM Output)

```

def create_circuit(params):
    circuit = Circuit('Single-Stage Amplifier')
    circuit.model('nmos_model', 'nmos', **nmos_params)
    # ...
    circuit.MOSFET('1', 'Vout', 'Vin', circuit.gnd,
                  circuit.gnd, model='nmos_model',
                  w=params['w_M1'], l=0.045e-6)
    circuit.R('1', 'Vout', 'Vdd', f"{params['r_load']}k")
    return circuit

param_ranges_definition = {
    'w_M1': {'min': 0.045e-6, 'max': 22.5e-6, 'log': True},
    'r_load': {'min': 5, 'max': 20, 'log': True},
}
initial_params = {'w_M1': 5e-6, 'r_load': 10}
    
```

With the tunable parameters, we use the Bayesian optimizer for circuit sizing. For each parameter set proposed by the optimizer, the input voltage bias is not treated as an optimization variable, because the output voltage is highly sensitive to the bias point: even small changes in input voltage can cause large variations in output voltage. Therefore, we propose a novel multi-resolution input bias search process to determine the optimal input bias ( $V_{in}$ ) before the main optimization loop. As illustrated in Fig. 7, this algorithm performs DC sweeps at progressively refined resolutions to locate the bias point that sets the output voltage close to  $V_{DD}/2$  in single-ended, single-supply configurations, providing a balanced operating point and placing the key transistors in their intended operating regions. Once this optimal bias is established, we proceed with a comprehensive circuit simulation to evaluate the figure of merit (FoM), which then provides the reward signal for the optimizer. This decoupling strategy reduces the dimensionality of the optimization space, improves convergence efficiency, and ensures proper circuit biasing for each parameter configuration under evaluation.

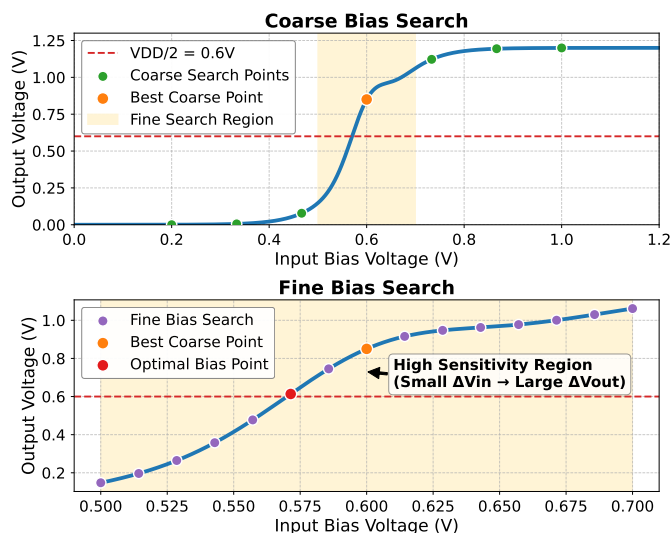


Fig. 7 Multi-Resolution Search for Input Bias. It conducts a coarse-to-fine DC sweep to automatically locate the input bias that sets the output voltage closest to  $V_{DD}/2$ , ensuring proper circuit biasing while reducing optimization dimensionality.

## IV. EXPERIMENTAL RESULTS

### A. Experimental settings

**Benchmark.** For topology generation, we consider 30 design tasks spanning 15 circuit types, including amplifiers, operational amplifiers, current mirrors, mixers, filters, and comparators, as summarized in TABLE III. Each circuit type is accompanied by functional verification code that validates the generated netlists through simulation results. The tasks are categorized into three difficulty levels (easy, medium, and hard) according to the number of components and the complexity of their interconnections. A circuit is considered composite when it involves the reuse of subcircuits. In our task design, Tasks 9 and 22–30 are composite circuits that utilize the circuit tool library, whereas all other tasks correspond to basic circuits. For circuit device sizing, we evaluated the ability of LLMs to generate performance-driven analog circuit topologies using 12 distinct design instructions (see Fig. 8).

**Metrics.** For topology generation, we adopt the widely-used  $\text{Pass}@k$ , which measures the probability that at least one correct solution appears in  $k$  sampled generations. Formally, it is defined as  $\text{Pass}@k = 1 - C_{n-c}^k / C_n^k$ , where  $n$  denotes the total number of attempts,  $c$  denotes the number of correct generations, and  $C_n^k$  denotes the binomial coefficient. In our experiments, we set  $n = 30$ . Note that we use the feedback-enhanced flow, allowing each generation task to be revised up to two additional times based on error information. For circuit device sizing, we align with AnalogGenie [35], using  $\text{FoM} = \text{GBW} \times C_L / \text{Power}$  as the primary optimization target. For completeness, we also evaluate Gain and GBW as alternative optimization goals.

**Hyper-parameters.** For topology generation, we set the temperature to 0.5 and `top_p` to 1.0 to encourage output diversity, while for parameter extraction, the temperature is fixed at 0.0. For circuit device sizing, we use Bayesian optimization with

TABLE III Benchmark Descriptions. Difficulties are distinguished by background colors (easy, medium, and hard).

| ID | Type           | Circuit Description                          | ID | Type            | Circuit Description                       |
|----|----------------|--|----|-----------------|---|
| 1  | Amplifier      | Common-source amp. with R load               | 16 | Opamp           | Op-amp with active current mirror loads   |
| 2  | Amplifier      | 3-stage common-source amplifier with R loads | 17 | Current Mirror  | Cascode current mirror                    |
| 3  | Amplifier      | Common-drain amp. with R load                | 18 | Opamp           | Common-source op-amp with R loads         |
| 4  | Amplifier      | Common-gate amp. with R load                 | 19 | Mixer           | A Gilbert cell mixer                      |
| 5  | Amplifier      | Cascode amp. with R load                     | 20 | Opamp           | Cascode op-amp with cascode loads         |
| 6  | Inverter       | NMOS inverter with R load                    | 21 | Opamp           | 2-stage op-amp with active loads          |
| 7  | Inverter       | Logical inverter                             | 22 | Oscillator      | Wien Bridge oscillator                    |
| 8  | Current Mirror | NMOS constant current source with R load     | 23 | Oscillator      | RC Shift oscillator                       |
| 9  | Comparator     | an op-amp comparator                         | 24 | Integrator      | Op-map integrator                         |
| 10 | Filter         | a passive low-pass filter                    | 25 | Differentiator  | Op-map differentiator                     |
| 11 | Filter         | a passive high-pass filter                   | 26 | Adder           | Op-map adder                              |
| 12 | Filter         | a passive band-pass filter                   | 27 | Subtractor      | Op-map subtractor                         |
| 13 | Filter         | a passive band-stop filter                   | 28 | Schmitt trigger | Non-inverting Schmitt trigger             |
| 14 | Amplifier      | Common-source amp. with diode-connected load | 29 | VCO             | Voltage-controlled Wien-bridge oscillator |
| 15 | Amplifier      | 2-stage amplifier with Miller compensation   | 30 | PLL             | Phase-locked loop                         |

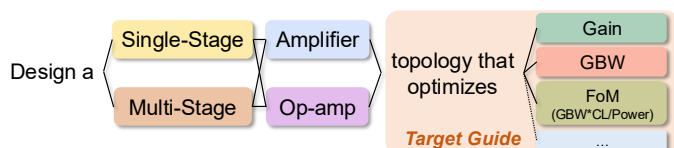


Fig. 8 Unified Generation and Optimization Prompt Strategies. All prompt variants illustrated here are evaluated in TABLE IX, with the target-guided objective (e.g., Gain, GBW, or user-defined FoM) being optional for topology optimization.

the Optuna package [93] (TPE sampler, 1000 trials), including 25% random initialization (250 trials) and multivariate sampling to account for parameter dependencies. The feedback flow allows up to three attempts, balancing token cost and success rate since most errors are corrected within this range. For multimodal LLMs, we use the native version of each model when available (e.g., GPT-4.1); otherwise, we employ alternative variants (GPT-4o for DeepSeek-V3, Qwen-VL for Qwen-Coder). All experiments are conducted with the BPTM 45nm PDK, consistent with AnalogGenie [36].

### B. Main Results

For the topology generation task, we evaluate the functional correctness of the generated netlists. Each task is attempted 30 times per model and counted as solved if at least one valid netlist is produced. TABLE IV provides a quantitative comparison of multiple LLMs evaluated in the same AnalogCoder-Pro framework under identical settings. The results indicate that the best proprietary models outperform open-source alternatives, with GPT-5 achieving the best performance (29/30 tasks, 95.3% Pass@5). Moreover, there is a clear positive correlation between general-purpose LLM capability and analog circuit generation performance, with more advanced models consistently achieving higher success rates across all circuit types.

### C. Unified Generation and Optimization

For each circuit type, we compared two prompt strategies: one including an explicit design objective (e.g., “Design a multi-stage op-amp topology that optimizes FoM (GBW\*CL/Power)”) and one omitting it (e.g., “Design a

multi-stage op-amp”), to evaluate the effectiveness of unified design. Optimization is performed with 1000 samples per trial, and the best value from each run is recorded. The unified generation and optimization process is repeated 30 times, and we report the best value among all 30 trials. As shown in TABLE IX, each base model is reported with two rows (without vs. with target-guided prompts), and the target-guided setting consistently improves objective-specific results, demonstrating that the LLM can effectively incorporate optimization goals during topology generation.

Beyond the analysis of unified prompt strategies, we further evaluate the performance of AnalogCoder-Pro in terms of FoM. According to TABLE IX, AnalogCoder, aided by multi-resolution input bias search, generates circuits that achieve competitive baseline performance even with only the initial LLM-assigned sizes and no further optimization. In contrast, AnalogCoder-Pro combines LLM-based parameter extraction with Bayesian optimization, yielding substantial gains over AnalogGenie, including up to  $2\times$  better op-amp FoM. To directly assess the role of multi-resolution input bias search, we compared it with several alternatives, including single-resolution search, fixed bias, and treating input bias as an optimizable parameter, using the best FoM netlist generated by Claude-3.7 (Fig. 10(a)). For each method, the optimizer was run using five random seeds, constraining the input bias range to  $0.25V_{dd}-0.75V_{dd}$  to avoid supply rail nonlinearities; variance across runs is shown as error bars. The single-resolution baseline employed 2000 uniform steps, while the multi-resolution strategy used 20, 200, and 2000 steps at coarse, medium, and fine stages. As shown in Fig. 9, the multi-resolution search consistently outperforms all baselines across every evaluation metric. Additional amplifier topologies generated by AnalogCoder-Pro are provided in Fig. 10(b) and (c).

### D. Ablation study

Our ablation study investigates the contribution of different components in the AnalogCoder-Pro framework. Specifically, we examine the effects of (i) having the LLM directly generate SPICE netlists instead of Python code, (ii) removing in-context

TABLE IV Main results. All LLMs are evaluated using the complete AnalogCoder-Pro framework.

| Model Task ID | Circuit Type    | Qwen-2.5-Coder-32B |        | GPT-4.1-nano |        | Gemini-2.5-Flash |           | DeepSeek-V3 |           | GPT-4.1   |           | Claude-3.7-Sonnet |           | GPT-5     |             |
|---------------|-----------------|--------------------|--------|--------------|--------|------------------|-----------|-------------|-----------|-----------|-----------|-------------------|-----------|-----------|-------------|
|               |                 | Pass@1             | Pass@5 | Pass@1       | Pass@5 | Pass@1           | Pass@5    | Pass@1      | Pass@5    | Pass@1    | Pass@5    | Pass@1            | Pass@5    | Pass@1    | Pass@5      |
| 1             | Amplifier       | 80.0               | 100.0  | 73.3         | 100.0  | 100.0            | 100.0     | 100.0       | 100.0     | 100.0     | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 2             | Amplifier       | 80.0               | 100.0  | 6.7          | 31.0   | 100.0            | 100.0     | 100.0       | 100.0     | 100.0     | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 3             | Amplifier       | 70.0               | 99.9   | 33.3         | 89.1   | 100.0            | 100.0     | 100.0       | 100.0     | 100.0     | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 4             | Amplifier       | 66.7               | 99.8   | 90.0         | 100.0  | 80.0             | 100.0     | 96.7        | 100.0     | 100.0     | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 5             | Amplifier       | 20.0               | 70.2   | 26.7         | 81.5   | 93.3             | 100.0     | 100.0       | 100.0     | 100.0     | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 6             | Inverter        | 100.0              | 100.0  | 86.7         | 100.0  | 100.0            | 100.0     | 100.0       | 100.0     | 100.0     | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 7             | Inverter        | 100.0              | 100.0  | 86.7         | 100.0  | 100.0            | 100.0     | 100.0       | 100.0     | 100.0     | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 8             | Current Mirror  | 73.3               | 100.0  | 30.0         | 85.7   | 76.7             | 100.0     | 100.0       | 100.0     | 100.0     | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 9             | Comparator      | 53.3               | 98.6   | 73.3         | 100.0  | 3.3              | 16.7      | 100.0       | 100.0     | 100.0     | 100.0     | 100.0             | 100.0     | 53.3      | 98.6        |
| 10            | Filter          | 80.0               | 100.0  | 93.3         | 100.0  | 96.7             | 100.0     | 100.0       | 100.0     | 96.7      | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 11            | Filter          | 13.3               | 53.8   | 86.7         | 100.0  | 100.0            | 100.0     | 100.0       | 100.0     | 90.0      | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 12            | Filter          | 16.7               | 62.7   | 13.3         | 53.8   | 100.0            | 100.0     | 86.7        | 100.0     | 80.0      | 100.0     | 100.0             | 100.0     | 96.7      | 100.0       |
| 13            | Filter          | 0.0                | 0.0    | 6.7          | 31.0   | 100.0            | 100.0     | 36.7        | 91.8      | 36.7      | 91.8      | 66.7              | 100.0     | 100.0     | 100.0       |
| 14            | Amplifier       | 43.3               | 95.7   | 36.7         | 91.8   | 26.7             | 81.5      | 100.0       | 100.0     | 100.0     | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 15            | Amplifier       | 30.0               | 85.7   | 66.7         | 99.8   | 100.0            | 100.0     | 100.0       | 100.0     | 100.0     | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 16            | Opamp           | 0.0                | 0.0    | 16.7         | 62.7   | 100.0            | 100.0     | 100.0       | 100.0     | 76.7      | 100.0     | 100.0             | 100.0     | 96.7      | 100.0       |
| 17            | Current Mirror  | 0.0                | 0.0    | 3.3          | 16.7   | 30.0             | 85.7      | 60.0        | 99.4      | 66.7      | 99.8      | 6.7               | 33.3      | 66.7      | 99.8        |
| 18            | Opamp           | 0.0                | 0.0    | 16.7         | 62.7   | 96.7             | 100.0     | 96.7        | 100.0     | 93.3      | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 19            | Mixer           | 20.0               | 70.2   | 6.7          | 31.0   | 3.3              | 16.7      | 66.7        | 99.8      | 50.0      | 97.9      | 100.0             | 100.0     | 96.7      | 100.0       |
| 20            | Opamp           | 6.7                | 31.0   | 0.0          | 0.0    | 100.0            | 100.0     | 76.7        | 100.0     | 90.0      | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 21            | Opamp           | 0.0                | 0.0    | 0.0          | 0.0    | 90.0             | 100.0     | 6.7         | 31.0      | 46.7      | 96.9      | 86.7              | 100.0     | 56.7      | 99.1        |
| 22            | Oscillator      | 16.7               | 62.7   | 16.7         | 62.7   | 66.7             | 99.8      | 10.0        | 43.4      | 46.7      | 96.9      | 46.7              | 98.1      | 16.7      | 62.7        |
| 23            | Oscillator      | 0.0                | 0.0    | 0.0          | 0.0    | 76.7             | 100.0     | 30.0        | 85.7      | 73.3      | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 24            | Integrator      | 13.3               | 53.8   | 80.0         | 100.0  | 100.0            | 100.0     | 96.7        | 100.0     | 96.7      | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 25            | Differentiator  | 30.0               | 85.7   | 40.0         | 94.0   | 100.0            | 100.0     | 100.0       | 100.0     | 100.0     | 100.0     | 100.0             | 100.0     | 100.0     | 100.0       |
| 26            | Adder           | 60.0               | 99.4   | 63.3         | 99.7   | 100.0            | 100.0     | 100.0       | 100.0     | 30.0      | 85.7      | 100.0             | 100.0     | 80.0      | 100.0       |
| 27            | Subtractor      | 0.0                | 0.0    | 26.7         | 81.5   | 93.3             | 100.0     | 93.3        | 100.0     | 73.3      | 100.0     | 100.0             | 100.0     | 96.7      | 100.0       |
| 28            | Schmitt trigger | 20.0               | 70.2   | 20.0         | 70.2   | 93.3             | 100.0     | 40.0        | 94.0      | 33.3      | 89.1      | 66.7              | 100.0     | 90.0      | 100.0       |
| 29            | VCO             | 0.0                | 0.0    | 0.0          | 0.0    | 23.3             | 76.4      | 30.0        | 85.7      | 13.3      | 53.8      | 80.0              | 100.0     | 86.7      | 100.0       |
| 30            | PLL             | 0.0                | 0.0    | 0.0          | 0.0    | 0.0              | 0.0       | 0.0         | 0.0       | 0.0       | 0.0       | 0.0               | 0.0       | 0.0       | 0.0         |
| Avg           |                 | 33.1               | 58.0   | 36.7         | 64.8   | 78.3             | 89.2      | 77.6        | 91.0      | 76.4      | 93.7      | <b>88.4</b>       | 94.4      | 87.9      | <b>95.3</b> |
| # Solved      |                 | 21                 | 21     | 25           | 25     | <b>29</b>        | <b>29</b> | <b>29</b>   | <b>29</b> | <b>29</b> | <b>29</b> | <b>29</b>         | <b>29</b> | <b>29</b> | <b>29</b>   |

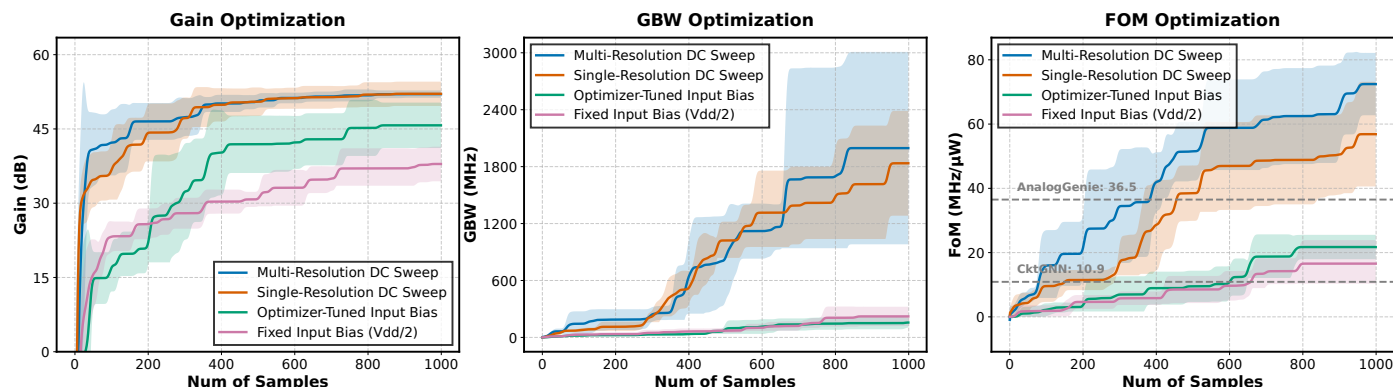


Fig. 9 Comparison of Input Bias Voltage Selection Methods. Each method is evaluated over 5 independent optimization runs. The proposed multi-resolution bias selection approach achieves faster and more accurate convergence to the optimal operating points, resulting in improved circuit performance metrics. The circuit topology for evaluation is shown in Fig. 10(a).

TABLE V Ablation Study Results

| Backbone     | Ablation                | Pass@1      | Pass@5      | # Solved  |
|--------------|-------------------------|-------------|-------------|-----------|
| GPT-4.1-nano | SPICE netlist           | 16.0        | 36.2        | 20        |
|              | w/o in-context learning | 27.1        | 50.7        | 23        |
|              | w/o chain-of-thought    | 29.0        | 53.9        | 22        |
|              | w/o feedback flow       | 17.6        | 42.0        | 19        |
|              | w/o library tool        | 26.0        | 44.6        | 18        |
|              | full framework          | <b>36.7</b> | <b>64.8</b> | <b>25</b> |
| DeepSeek-V3  | SPICE netlist           | 70.3        | 85.3        | 28        |
|              | w/o in-context learning | 55.2        | 76.7        | 28        |
|              | w/o chain-of-thought    | 71.0        | 86.2        | 27        |
|              | w/o feedback flow       | 68.0        | 80.3        | 28        |
|              | w/o library tool        | 62.0        | 70.5        | 25        |
|              | full framework          | <b>77.6</b> | <b>91.0</b> | <b>29</b> |

TABLE VI Token Consumption (Schmitt trigger design)

| Revision Times   | 0    | 1    | 2    |
|------------------|------|------|------|
| Success Rate (%) | 26.7 | 30.0 | 40.0 |
| Avg. Token       | 1578 | 4567 | 8207 |
| Avg. MLLM Token  | 0    | 1076 | 2224 |

TABLE VII Ablation Study of Bayesian Optimization

| Circuit Category   | FoM (w/o BO) | FoM (w/ BO) | BO Time (s) |
|--------------------|--------------|-------------|-------------|
| Single-Stage Amp   | 0.97         | 1.88        | 34.9        |
| Single-Stage Opamp | 1.08         | 1.60        | 54.3        |
| Multi-Stage Amp    | 0.17         | 13.24       | 45.5        |
| Multi-Stage Opamp  | 0.00         | 8.33        | 45.3        |

TABLE VIII Comparison of Multimodal Feedback-Enhanced Flow. Each model is evaluated with and without Multimodal LLM assistance (with • denoting assistance) in average Pass@1. The Avg. reports the mean score across categories. Circuit type abbreviations follow TABLE II.

| Model       | Multimodal | Osc  | Int   | Diff  | Sch  | Comp  | Mix   | Filt | Avg.        |
|-------------|------------|------|-------|-------|------|-------|-------|------|-------------|
| Qwen2.5     |            | 1.7  | 6.7   | 40.0  | 20.0 | 20.0  | 16.7  | 16.7 | 17.4        |
| -Coder-32B  | •          | 8.3  | 13.3  | 30.0  | 20.0 | 53.3  | 20.0  | 27.5 | <b>24.6</b> |
| GPT-4.1     |            | 43.3 | 93.3  | 96.7  | 36.7 | 93.3  | 43.3  | 75.0 | 68.8        |
|             | •          | 60.0 | 96.7  | 100.0 | 33.3 | 100.0 | 50.0  | 75.8 | <b>73.7</b> |
| DeepSeek-V3 |            | 8.3  | 100.0 | 96.7  | 50.0 | 100.0 | 63.3  | 77.5 | 70.8        |
|             | •          | 20.0 | 96.7  | 100.0 | 40.0 | 100.0 | 66.7  | 80.8 | <b>72.0</b> |
| Claude-3.7  |            | 70.0 | 100.0 | 100.0 | 26.7 | 100.0 | 100.0 | 91.7 | 84.1        |
| -Sonnet     | •          | 73.3 | 100.0 | 100.0 | 66.7 | 100.0 | 100.0 | 91.7 | <b>90.2</b> |

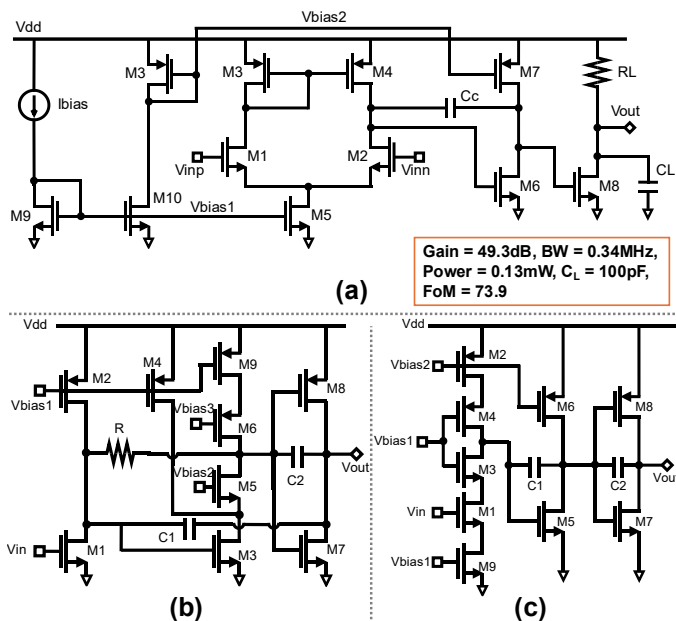


Fig. 10 Topologies generated by AnalogCoder-Pro with target-guided prompts. Functional topologies are obtained within 3 attempts and sized via 1000 optimization steps. (a) Op-amp with the highest FoM, also evaluated in Fig. 9; (b), (c) Amplifiers generated with target-guided design prompts.

learning from the design prompts, (iii) removing chain-of-thought guidance from the design prompts, (iv) excluding the feedback-enhanced workflow, and (v) omitting the circuit tool library, in comparison with (vi) the full framework. The results in TABLE V show that removing any component reduces performance. To further evaluate the role of MLLM, we compare the results with and without multimodal waveform analysis in the feedback flow. As shown in TABLE VIII, using MLLM consistently improves circuit topology generation success rate. We further evaluate the effect of Bayesian optimization (BO). For each circuit category, we take the first successfully generated LLM topology with Claude-Sonnet-3.5 and compare its FoM with and without BO. As shown in TABLE VII, BO consistently improves FoM with only moderate runtime overhead (tested with 16-core multiprocessing). The token usage is reported in TABLE VI (LLM: DeepSeek-V3, MLLM: GPT-4o).

### E. Discussion

The experimental results demonstrate that AnalogCoder-Pro effectively bridges the gap between general-purpose LLM reasoning and domain-specific analog design automation. Across models, we observe that stronger general-purpose LLMs (e.g., GPT-5 compared to GPT-4.1) also achieve higher success rates in analog circuit generation (TABLE IV). In error correction, the ablation study (TABLE V) confirms that our feedback-enhanced flow guides iterative repairs, while the inclusion of waveform visuals serves as a vital supplement to further boost debugging capabilities (TABLE VIII). Finally, by integrating target-guided topology generation with Bayesian Optimization (TABLE VII and TABLE IX), the proposed framework enables LLMs to proactively align architectural choices with performance objectives, improving the Figure of Merit (FoM) with minimal computational overhead.

## V. CONCLUSION

In this work, we introduce AnalogCoder-Pro, a unified, end-to-end multimodal LLM framework for automated analog circuit design. By combining multimodal diagnosis-and-repair mechanisms with a reusable circuit tool library and integrated performance-driven topology generation, AnalogCoder-Pro achieves state-of-the-art performance and substantially improves success rates across diverse circuit design tasks.

## REFERENCES

- [1] 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*, 2023.
- [2] R. Zhong, X. Du, S. Kai, Z. Tang, S. Xu, H.-L. Zhen, J. Hao, Q. Xu, M. Yuan, and J. Yan, “Llm4eda: Emerging progress in large language models for electronic design automation,” *arXiv*, 2023.
- [3] J. Pan, G. Zhou, C.-C. Chang, I. Jacobson, J. Hu, and Y. Chen, “A survey of research in large language models for electronic design automation,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 30, no. 3, pp. 1–21, 2025.
- [4] W. Fang, J. Wang, Y. Lu, S. Liu, Y. Wu, Y. Ma, and Z. Xie, “A survey of circuit foundation model: Foundation ai models for vlsi circuit design and eda,” *arXiv*, 2025.
- [5] J. Blocklove, S. Garg, R. Karri, and H. Pearce, “Chip-chat: Challenges and opportunities in conversational hardware design,” in *Workshop on Machine Learning for CAD (MLCAD)*. IEEE, 2023, pp. 1–6.
- [6] K. Chang, Y. Wang, H. Ren, M. Wang, S. Liang, Y. Han, H. Li, and X. Li, “Chippgt: How far are we from natural language hardware design,” *arXiv*, 2023.
- [7] S. Thakur, B. Ahmad, H. Pearce, B. Tan, B. Dolan-Gavitt, R. Karri, and S. Garg, “Verigen: A large language model for verilog code generation,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2023.
- [8] S. Thakur, J. Blocklove, H. Pearce, B. Tan, S. Garg, and R. Karri, “Autochip: Automating hdl generation using llm feedback,” *arXiv*, 2023.
- [9] Y. Fu, Y. Zhang, Z. Yu, S. Li, Z. Ye, C. Li, C. Wan, and Y. C. Lin, “Gpt4aigchip: Towards next-generation ai accelerator design automation via large language models,” in *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–9.
- [10] Y. Tsai, M. Liu, and H. Ren, “Rtlfixer: Automatically fixing rtl syntax errors with large language models,” *arXiv*, 2023.
- [11] Z. Pei, H.-L. Zhen, M. Yuan, Y. Huang, and B. Yu, “Betternv: Controlled verilog generation with discriminative guidance,” *arXiv*, 2024.
- [12] X. Yao, H. Li, T. H. Chan, W. Xiao, M. Yuan, Y. Huang, L. Chen, and B. Yu, “Hdldebugger: Streamlining hdl debugging with large language models,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2024.

TABLE IX Performance Comparison for Unified Circuit Generation and Optimization. For each instruction in Fig. 8, 30 netlist generation attempts are performed. Each correct netlist is optimized with 1000 samples, and the best performance among the 30 trials is reported. Maximum values are in **bold**. AnalogGenie and CktGNN results are from the original papers.

| Method<br># of Simulations | Base Model  | Target-guided <sup>2</sup> | Single-Stage Amp |              |                  | Single-Stage Opamp |              |             | Multi-Stage Amp |                |              | Multi-Stage Opamp |               |             | Avg. Rank |     |
|----------------------------|-------------|----------------------------|------------------|--------------|------------------|--------------------|--------------|-------------|-----------------|----------------|--------------|-------------------|---------------|-------------|-----------|-----|
|                            |             |                            | Gain             | GBW          | FoM <sup>1</sup> | Gain               | GBW          | FoM         | Gain            | GBW            | FoM          | Gain              | GBW           | FoM         |           |     |
| CktGNN [84]                | GNN         | •                          | -                | -            | -                | -                  | -            | -           | -               | -              | -            | -                 | -             | -           | 10.9      | -   |
| AnalogGenie(>1k) [35]      | GPT-2       | •                          | -                | -            | -                | -                  | -            | -           | -               | -              | -            | -                 | -             | -           | 36.5      | -   |
| AnalogCoder(1) [27]        | GPT-4o      | •                          | 13.9             | 93.2         | 1.7              | 12.8               | 113.1        | 0.6         | 40.6            | 2076.2         | 5.7          | 18.0              | 154.6         | 0.5         | 10.5      | 9.1 |
|                            |             |                            | 16.2             | 191.7        | 1.1              | 13.4               | 85.1         | 0.8         | 43.8            | 4195.4         | 8.8          | 35.5              | 880.8         | 2.4         |           |     |
|                            | DeepSeek-V3 | •                          | 0.7              | 23.6         | 1.7              | 13.5               | 147.7        | 0.9         | 37.5            | 5330.6         | 8.5          | 27.8              | 1050.9        | 4.1         | 9.4       | 7.6 |
|                            |             |                            | 26.2             | 198.5        | 1.0              | 29.7               | 153.5        | 2.6         | 37.5            | 9808.7         | 13.1         | 7.8               | 443.1         | 4.2         |           |     |
| Claude-3.7                 | •           | •                          | 16.0             | 49.4         | 1.9              | 14.5               | 130.0        | 1.1         | 35.8            | 314.8          | 1.2          | 67.9              | 2384.8        | 12.1        | 8.2       | 7.9 |
|                            |             |                            | 30.5             | 157.4        | 1.0              | 23.1               | 13.7         | 1.1         | 0.0             | 4081.3         | 165.9        | 40.0              | 899.0         | 0.5         |           |     |
| AnalogCoder-Pro(1k)        | GPT-4o      | •                          | 18.6             | 416.8        | 2.0              | 20.6               | 299.1        | 2.9         | 56.3            | 23111.9        | 133.2        | 33.3              | 1406.1        | 9.5         | 4.9       | 4.5 |
|                            |             |                            | 18.9             | <b>443.3</b> | 2.4              | 20.4               | 260.3        | 3.6         | 55.7            | 35873.3        | 124.0        | 50.2              | 1122.2        | 37.4        |           |     |
|                            | DeepSeek-V3 | •                          | 16.1             | 50.4         | 1.9              | 20.5               | 279.9        | 3.0         | 55.9            | 22579.2        | 127.3        | 36.4              | 2346.0        | 14.1        | 5.7       | 2.5 |
|                            |             |                            | 27.5             | 418.8        | 2.5              | 36.2               | <b>363.3</b> | <b>14.5</b> | <b>56.4</b>     | <b>41316.7</b> | 185.7        | 36.6              | 2243.9        | 36.2        |           |     |
| Claude-3.7                 | •           | •                          | 16.1             | 100.6        | 1.9              | 27.8               | 281.3        | 8.5         | 53.2            | 5298.2         | 135.1        | 67.9              | <b>6279.4</b> | 43.4        | 4.3       | 2.6 |
|                            |             |                            | <b>45.3</b>      | 413.3        | <b>3.2</b>       | <b>39.2</b>        | 266.6        | 7.4         | 46.1            | 17366.0        | <b>413.2</b> | <b>74.9</b>       | 3193.0        | <b>73.9</b> |           |     |

<sup>1</sup> Gain (dB) =  $20 \times \log_{10}(V_{out}/V_{in})$ , GBW (MHz) = Gain (linear)  $\times$  Bandwidth, FoM (MHz-pF/ $\mu$ W) = GBW  $\times$   $C_L$ /Power, where  $C_L = 100$  pF.

<sup>2</sup> Target-guided design indicates that specific design objectives (e.g., maximizing the Figure of Merit, FoM) were set for the LLM, as shown in Fig. 8.

- [13] Y. Zhu, D. Huang, H. Lyu, X. Zhang, C. Li, W. Shi, Y. Wu, J. Mu, J. Wang, Y. Zhao *et al.*, "Codev-r1: Reasoning-enhanced verilog generation," *arXiv*, 2025.
- [14] M. Akyash, K. Azar, and H. Kamali, "Rtl++: Graph-enhanced llm for rtl code generation," *arXiv*, 2025.
- [15] M. Liu, Y.-D. Tsai, W. Zhou, and H. Ren, "Craftrtl: High-quality synthetic data generation for verilog code models with correct-by-construction non-textual representations and targeted code repair," *arXiv*, 2024.
- [16] C. Deng, Y.-D. Tsai, G.-T. Liu, Z. Yu, and H. Ren, "Scalertl: Scaling llms with reasoning data and test-time compute for accurate rtl code generation," *arXiv*, 2025.
- [17] S. Liu, W. Fang, Y. Lu, J. Wang, Q. Zhang, H. Zhang, and Z. Xie, "Rtlcoder: Fully open-source and efficient llm-assisted rtl code generation technique," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2024.
- [18] Y. Zhao, D. Huang, C. Li, P. Jin, M. Song, Y. Xu, Z. Nan, M. Gao, T. Ma, L. Qi *et al.*, "Codev: Empowering llms with hdl generation through multi-level summarization," *arXiv*, 2024.
- [19] M. Liu, T.-D. Ene, R. Kirby, C. Cheng, N. Pinckney, R. Liang, J. Alben, H. Anand, S. Banerjee, I. Bayraktaroglu *et al.*, "Chipnemo: Domain-adapted llms for chip design," *arXiv*, 2023.
- [20] H. Wu, Z. He, X. Zhang, X. Yao, S. Zheng, H. Zheng, and B. Yu, "Chateda: A large language model powered autonomous agent for eda," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (TCAD)*, 2024.
- [21] H. Wu, H. Zheng, Z. He, and B. Yu, "Divergent thoughts toward one goal: Llm-based multi-agent collaboration system for electronic design automation," *arXiv*, 2025.
- [22] Y. Pu, Z. He, T. Qiu, H. Wu, and B. Yu, "Customized retrieval augmented generation and benchmarking for eda tool documentation qa," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design (TODAES)*, 2024, pp. 1–9.
- [23] C. Liu, Y. Liu, Y. Du, and L. Du, "Ladac: Large language model-driven auto-designer for analog circuits," *Authorea Preprints*, 2024.
- [24] Y. Yin, Y. Wang, B. Xu, and P. Li, "Ado-llm: Analog design bayesian optimization with in-context learning of large language models," in *International Conference on Computer-Aided Design (ICCAD)*, 2024.
- [25] C.-C. Chang, Y. Shen, S. Fan, J. Li, S. Zhang, N. Cao, Y. Chen, and X. Zhang, "Lamagic: Language-model-based topology generation for analog integrated circuits," *arXiv*, 2024.
- [26] C.-C. Chang, W.-H. Lin, Y. Shen, Y. Chen, and X. Zhang, "Lamagic2: Advanced circuit formulations for language model-based analog topology generation," in *International Conference on Machine Learning (ICML)*, 2025.
- [27] 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 *AAAI Conference on Artificial Intelligence (AAAI)*, 2025.
- [28] D. Vungarala, S. Alam, A. Ghosh, and S. Angizi, "Spicepilot: Navigating spice code generation and simulation with ai guidance," *arXiv*, 2024.
- [29] D. V. Kochar, H. Wang, A. Chandrakasan, and X. Zhang, "Ledro: Llm-enhanced design space reduction and optimization for analog circuits," *arXiv*, 2024.
- [30] Z. Chen, J. Huang, Y. Liu, F. Yang, L. Shang, D. Zhou, and X. Zeng, "Artisan: Automated operational amplifier design via domain-specific large language model," in *Design Automation Conference (DAC)*, 2024.
- [31] 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*, 2024.
- [32] 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," *Authorea Preprints*, 2024.
- [33] H. Zhang, S. Sun, Y. Lin, R. Wang, and J. Bian, "Analogxpert: Automating analog topology synthesis by incorporating circuit design expertise into large language models," *arXiv*, 2024.
- [34] J. Bhandari, V. Bhat, Y. He, H. Rahmani, S. Garg, and R. Karri, "Masalachai: A large-scale spice netlist dataset for analog circuits by harnessing ai," 2025.
- [35] J. Gao, W. Cao, J. Yang, and X. Zhang, "Analoggenie: A generative engine for automatic discovery of analog circuit topologies," in *International Conference on Learning Representations (ICLR)*, 2025.
- [36] J. Gao, W. Cao, and X. Zhang, "Analoggenie-lite: Enhancing scalability and precision in circuit topology discovery through lightweight graph modeling," in *International Conference on Machine Learning (ICML)*, 2025.
- [37] Q. Li, S. Hong, J. Gao, X. Zhang, T. Lan, and W. Cao, "Analogfed: Federated discovery of analog circuit topologies with generative ai," 2025.
- [38] P. Vijayaraghavan, L. Shi, E. Degan, V. Mukherjee, and X. Zhang, "Autocircuit-rl: Reinforcement learning-driven llm for automated circuit topology generation," in *International Conference on Machine Learning (ICML)*, 2025.
- [39] S. Nau, J. Krummenauer, and A. Zimmermann, "Spiceassistant: Llm using spice simulation tools for schematic design of switched-mode power supplies," *arXiv preprint arXiv:2507.10639*, 2025.
- [40] P.-H. Chen, Y.-S. Lin, W.-C. Lee, T.-Y. Leu, P.-H. Hsu, A. Dissanayake, S. Oh, and C.-S. Chiu, "Menter: A fully-automated multi-agent workflow for end-to-end rf/analog circuits netlist design," in *IEEE International Conference on LLM-Aided Design (ICLAD)*, 2025.
- [41] B. Razavi, *Design of Analog CMOS Integrated Circuits*. McGraw-Hill, Inc., 2000.
- [42] G. Stehr, H. Graeb, and K. Antreich, "Performance trade-off analysis of analog circuits by normal-boundary intersection," in *Design Automation Conference (DAC)*, 2003, pp. 958–963.

- [43] D. Mueller-Gritschneider, H. Graeb, and U. Schlichtmann, "A successive approach to compute the bounded pareto front of practical multiobjective optimization problems," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 915–934, 2009.
- [44] H. Chae, S. Kim, S. Poddar, X. Gao, S. Li, and D. Z. Pan, "Towards Generative AI for Analog and RF IC Design: From Spec to Layout," in *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2025, pp. 1–9.
- [45] X. Yang, J. Gao, Y. Wang, and X. Zhang, "ZeroSim: Zero-Shot Analog Circuit Evaluation with Unified Transformer Embeddings," in *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2025, pp. 1–9.
- [46] D. Thomas and P. Moorby, *The Verilog® hardware description language*. Springer Science & Business Media, 2008.
- [47] D. Guo, Q. Zhu, D. Yang, Z. Xie, K. Dong, W. Zhang, G. Chen, X. Bi, Y. Wu, Y. Li *et al.*, "Deepseek-coder: When the large language model meets programming—the rise of code intelligence," *arXiv*, 2024.
- [48] A. Vladimirescu, *The SPICE book*. John Wiley & Sons, Inc., 1994.
- [49] S. Poddar, C.-T. Ho, Z. Wei, W. Cao, H. Ren, and D. Z. Pan, "HearRT: A Hierarchical Circuit Reasoning Tree-Based Agentic Framework for AMS Design Optimization," *arXiv preprint arXiv:2511.19669*, 2025.
- [50] H. Wang, K. Yang, 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 *Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.
- [51] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolic, "Autocckt: Deep reinforcement learning of analog circuit designs," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020.
- [52] 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 *Design Automation Conference (DAC)*, 2021.
- [53] A. F. Budak, D. Smart, B. Swahn, and D. Z. Pan, "Apostle: Asynchronously parallel optimization for sizing analog transistors using dnn learning," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2023.
- [54] M. Choi, Y. Choi, K. Lee, and S. Kang, "Reinforcement learning-based analog circuit optimizer using g m/i d for sizing," in *Design Automation Conference (DAC)*. IEEE, 2023.
- [55] Y. Li, Y. Lin, M. Madhusudan, A. Sharma, S. Sapatnekar, R. Harjani, and J. Hu, "A circuit attention network-based actor-critic learning approach to robust analog transistor sizing," in *Workshop on Machine Learning for CAD (MLCAD)*, 2021.
- [56] A. F. Budak, K. Zhu, H. Chen, S. Poddar, L. Zhao, Y. Jia, and D. Z. Pan, "Joint Optimization of Sizing and Layout for AMS Designs: Challenges and Opportunities," in *Proceedings of the 2023 International Symposium on Physical Design*, 2023, pp. 84–92.
- [57] Y. Oh, D. Kim, Y. H. Lee, and B. Hwang, "Cronus: Circuit rapid optimization with neural simulator," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024.
- [58] S. Poddar, Y. Oh, Y. Lai, H. Zhu, B. Hwang, and D. Z. Pan, "INSIGHT: A Universal Neural Simulator Framework for Analog Circuits with Autoregressive Transformers," in *Design Automation Conference (DAC)*, 2025.
- [59] B. Liu, Y. Wang, Z. Yu, L. Liu, M. Li, Z. Wang, J. Lu, and F. V. Fernández, "Analog circuit optimization system based on hybrid evolutionary algorithms," *Integration*, vol. 42, no. 2, 2009.
- [60] W. Yu and X. Wang, *Advanced field-solver techniques for RC extraction of integrated circuits*. Springer, 2014.
- [61] T. McConaghy, P. Palmers, G. Gielen, and M. Steyaert, "Simultaneous multi-topology multi-objective sizing across thousands of analog circuit topologies," in *Design Automation Conference (DAC)*, 2007.
- [62] P. Palmers, T. McConaghy, M. Steyaert, and G. Gielen, "Massively multi-topology sizing of analog integrated circuits," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2009.
- [63] P. Veselinovic, D. Leenaerts, W. Van Bokhoven, F. Leyn, F. Proesmans, G. Gielen, and W. Sansen, "A flexible topology selection program as part of an analog synthesis system," in *European Design and Test Conference*, 1995.
- [64] T. McConaghy, P. Palmers, G. Gielen, and M. Steyaert, "Automated extraction of expert knowledge in analog topology selection and sizing," in *International Conference on Computer-Aided Design (ICCAD)*, 2008.
- [65] Z. Zhao and L. Zhang, "An automated topology synthesis framework for analog integrated circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (TCAD)*, vol. 39, no. 12, pp. 4325–4337, 2020.
- [66] P. C. Maulik, L. R. Carley, and R. A. Rutenbar, "Integer programming based topology selection of cell-level analog circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 14, no. 4, pp. 401–412, 1995.
- [67] J. Lu, Y. Li, F. Yang, L. Shang, and X. Zeng, "High-level topology synthesis method for  $\delta$ - $\sigma$  modulators via bi-level bayesian optimization," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 70, no. 12, pp. 4389–4393, 2023.
- [68] S. Poddar, A. Budak, L. Zhao, C.-H. Hsu, S. Maji, K. Zhu, Y. Jia, and D. Z. Pan, "A Data-Driven Analog Circuit Synthesizer with Automatic Topology Selection and Sizing," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024.
- [69] B. Liu, H. Zhang, X. Gao, Z. Kong, X. Tang, Y. Lin, R. Wang, and R. Huang, "Layoutcopilot: An llm-powered multi-agent collaborative framework for interactive analog layout design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2025.
- [70] C. Liu, W. Chen, H. Xu, Y. Du, J. Yang, and L. Du, "A large language model-based multi-agent framework for analog circuits' sizing relationships extraction," *arXiv preprint arXiv:2506.18424*, 2025.
- [71] A. Hammoud, C. Goyal, S. Pathen, A. Dai, A. Li, G. Kielian, and M. Saligane, "Human language to analog layout using layout layout automation framework," in *International Symposium on Machine Learning for CAD (MLCAD)*, 2024.
- [72] 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 *International Symposium of Electronics Design Automation (ISED)*. IEEE, 2025, pp. 201–207.
- [73] P. Abbineni, S. Aldowaish, C. Liechty, S. Noorzad, A. Ghazizadeh, and M. Fayazi, "Muallm: A multimodal large language model agent for circuit design assistance with hybrid contextual retrieval-augmented generation," *arXiv*, 2025.
- [74] C. Zhao, Z. Shi, X. Wen, C. Liu, Y. Liu, Y. Zhou, Y. Zhao, H. Feng, Y. Zhu, G.-W. Wan *et al.*, "Mmcircuiteval: A comprehensive multimodal circuit-focused benchmark for evaluating llms," *arXiv preprint arXiv:2507.19525*, 2025.
- [75] Y. Shi, Z. Zhang, H. Wang, Z. Tao, Z. Li, B. Chen, Y. Wang, Z. Yu, T.-J. Lin, and L. He, "Amsbench: A comprehensive benchmark for evaluating mllm capabilities in ams circuits," *arXiv preprint arXiv:2505.24138*, 2025.
- [76] Z. Chen, J. Zhuang, J. Shen, X. Ke, X. Yang, M. Zhou, Z. Du, X. Yan, Z. Wu, Z. Xu *et al.*, "Analogseeker: An open-source foundation language model for analog circuit design," *arXiv*, 2025.
- [77] H. Chen, M. Liu, B. Xu, K. Zhu, X. Tang, S. Li, Y. Lin, N. Sun, and D. Z. Pan, "Magical: An open-source fully automated analog ic layout system from netlist to gdsii," *IEEE Design & Test*, 2020.
- [78] S. Maji, A. F. Budak, S. Poddar, and D. Z. Pan, "Toward End-to-End Analog Design Automation with ML and Data-Driven Approaches," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2024, pp. 657–664.
- [79] S. Maji, H. Park, G.-M. Hong, S. Poddar, and D. Z. Pan, "Multiobjective Optimization for Common-Centroid Placement of Analog Transistors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 44, no. 6, pp. 2029–2039, 2024.
- [80] G. Chen, K. Zhu, S. Kim, H. Zhu, Y. Lai, B. Yu, and D. Z. Pan, "Llm-enhanced bayesian optimization for efficient analog layout constraint generation," *arXiv*, 2024.
- [81] P. Mangalagiri, L. Qian, F. Zafar, P. Mosalikanti, P. Chang, A. Kurian, and V. Saripalli, "Cdls: Constraint driven generative ai framework for analog layout synthesis," in *Design Automation Conference (DAC)*, 2024.
- [82] Z. Zhao and L. Zhang, "Deep reinforcement learning for analog circuit structure synthesis," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022.
- [83] 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," *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2025.
- [84] Z. Dong, W. Cao, M. Zhang, D. Tao, Y. Chen, and X. Zhang, "Cktgmn: Circuit graph neural network for electronic design automation," in *International Conference on Learning Representations (ICLR)*, 2023.

- [85] Zhao, Zhenxin and Zhang, Lihong, "Graph-grammar-based analog circuit topology synthesis," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.
- [86] C. Liu, J. Li, Y. Feng, W. Huang, W. Chen, Y. Du, J. Yang, and L. Du, "Diffckt: A diffusion model-based hybrid neural network framework for automatic transistor-level generation of analog circuits," *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2025.
- [87] F. Salvaire, "Pyspice," 2021.
- [88] Q. Zheng, X. Xia, X. Zou, Y. Dong, S. Wang, Y. Xue, L. Shen, Z. Wang, A. Wang, Y. Li *et al.*, "Codegeex: A pre-trained model for code generation with multilingual benchmarking on humaneval-x," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2023, pp. 5673–5684.
- [89] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, and Z. Sui, "A survey on in-context learning," *arXiv*, 2022.
- [90] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems (NeurIPS)*, vol. 35, pp. 24 824–24 837, 2022.
- [91] G. Wang, Y. Xie, Y. Jiang, A. Mandlkar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," in *Intrinsically-Motivated and Open-Ended Learning Workshop@ NeurIPS2023*, 2023.
- [92] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian *et al.*, "Toollm: Facilitating large language models to master 16000+ real-world apis," *arXiv*, 2023.
- [93] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *ACM SIGKDD international conference on knowledge discovery & data mining (SIGKDD)*, 2019, pp. 2623–2631.



**Yao Lai** received the Ph.D. degree in computer science from The University of Hong Kong in 2025, where he was a member of the HKU-MMLab. He received the B.Eng. degree from Fudan University and the M.Eng. degree from Tsinghua University. His research interests include artificial intelligence for electronic design automation (AI for EDA) and multimodal large language models. His paper is recognized as one of the most influential AAAI 2025 papers.



**Souradip Poddar** received the B.Tech. degree in Electrical Engineering from the Indian Institute of Technology Kharagpur, India, in 2019, graduating with Department Rank 1. He received the M.S.E. degree in Electrical and Computer Engineering from The University of Texas at Austin in 2025, where he is currently pursuing the Ph.D. degree. His research focuses on leveraging advancements in machine learning to enhance VLSI design automation workflows, particularly for mixed-signal circuits.



**Sungyoung Lee** is a Ph.D. student in computer science at The University of Texas at Austin. He received his B.S. degree in Electrical Engineering and Computer Science from Seoul National University, South Korea, in 2023. His research interests include reinforcement learning and large-scale distributed systems. He has received the DAC Young Fellowship and the UT Excellence Engineering Fellowship.



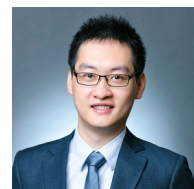
**Guojin Chen** earned his Ph.D. in Computer Science and Engineering at The Chinese University of Hong Kong (CUHK), advised by Prof. Bei Yu. He also holds an M.S. from CUHK and a B.S. from Huazhong University of Science and Technology (HUST). His research interests include agentic large language models, computational lithography, and optimization for electronic design automation (EDA).



**Mengkang Hu** is a third-year Ph.D. student at The University of Hong Kong, affiliated with HKU-MMLab. His research interests include language agents and embodied AI. He has published 20+ papers in international conferences such as ICML, ICLR, NeurIPS, with over 1200 citations on Google Scholar. He led the development of OWL, a multi-agent assistance system for real-world tasks, which has received more than 18.4k stars on GitHub.



**Bei Yu** (M'15-SM'22) received the Ph.D. degree from The University of Texas at Austin in 2014. He is currently a Professor in the Department of Computer Science and Engineering, The Chinese University of Hong Kong. He has served as TPC Chair of ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees. He received eleven Best Paper Awards from ICCAD 2024 & 2021 & 2013, IEEE TSM 2022, DATE 2022, ASPDAC 2021 & 2012, ICTAI 2019, Integration, the VLSI Journal in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, six ICCAD/ISPD contest awards, IEEE CEDA Ernest S. Kuh Early Career Award in 2021, DAC Under-40 Innovator Award in 2024, and Hong Kong RGC Research Fellowship Scheme (RFS) Award in 2024.



**Ping Luo** is an Associate Professor in the Department of Computer Science at the University of Hong Kong, an Associate Director of the HKU Musketeers Foundation Institute of Data Science (HKU IDS), and a Deputy Director of the Joint Research Lab of HKU and Shanghai AI Lab. He obtained his Ph.D. in Information Engineering from the Chinese University of Hong Kong in 2014, under the supervision of Prof. Xiaou Tang (founder of SenseTime) and Prof. Xiaogang Wang. Before joining HKU in 2019, he was a Research Director at SenseTime. He has published 200+ papers in international conferences and journals such as TPAMI, ICML, ICLR, NeurIPS, and CVPR, with over 100,000 citations on Google Scholar. He was awarded the 2015 AAAI Easily Accessible Paper, nominated for the 2022 Computational Visual Media Journal's Best Paper of the Year, won the 2022 ACL Outstanding Paper, the 2022, 2023, 2025 World Artificial Intelligence Conference (WAIC) Outstanding Papers, and was a candidate for the Best Paper at ICCV'23. He was recognized as one of the innovators under 35 in the Asia-Pacific region by the MIT Technology Review (MIT TR35) in 2020. He has mentored 30 Ph.D. students, many of whom have received significant awards such as the Nvidia Fellowship, Baidu Fellowship, ByteDance Fellowship, WAIC Yunfan Award, Tencent Project Up Scholarship, etc.



**David Z. Pan** (M'00-SM'06-F'14) received the B.S. degree from Peking University in 1992 and the M.S. and Ph.D. degrees from the University of California, Los Angeles, in 1994, 1998, and 2000, respectively. From 2000 to 2003, he was a Research Staff Member at the IBM Thomas J. Watson Research Center. He is currently a Full Professor and the Silicon Laboratories Endowed Chair of Electrical Engineering at The University of Texas at Austin. His research interests include design automation for digital/analog/mixed-signal/RF ICs and emerging technologies, synergistic AI/IC co-optimization, domain-specific accelerators, design for manufacturing, and hardware security. He has published over 500 peer-reviewed papers and holds nine U.S. patents. He has supervised more than 50 Ph.D. students and Postdoctoral Fellows. He has received numerous awards, including the SRC Technical Excellence Award, 21 Best Paper Awards, the ACM/SIGDA Outstanding New Faculty Award, the NSF CAREER Award, the IBM Faculty Award (four times), the CACM Research Highlights recognition, and several dissertation and student research awards with his students. He has served as Senior Associate Editor or Associate Editor for multiple IEEE and ACM journals and held leadership roles in major conferences, including Program Chair and General Chair for ICCAD, ASP-DAC, and ISPD. He is also a Fellow of ACM and SPIE.