SAAC: SIZING AGENT FOR ANALOG CIRCUIT VIA TURN-LEVEL REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Analog circuit sizing (ACS) is a critical yet challenging iterative optimization task in electronic design automation (EDA), requiring domain expertise, multiple expensive simulations, and the balancing of conflicting performance objectives. Existing optimization approaches face distinct limitations: traditional black-box methods like Bayesian Optimization ignore the reasoning process and prior knowledge, relying instead on extensive sampling that scales poorly with design complexity. While large language models (LLMs) offer promising reasoning capabilities, current LLM-based ACS workflows either depend heavily on the base model's inherent abilities or require costly human-annotated data for training, and existing RL-based methods also fail to perform fine-grained, turn-level optimization at each iteration. To address these gaps, we propose SAAC (**S**izing **A**gent for **A**nalog **C**ircuit), an agentic framework specifically tailored for analog circuit sizing. SAAC integrates a professional-grade simulation environment that provides standardized tool interactions and verifiable reward signals, coupled with an automated data synthesis pipeline that eliminates the need for expert-annotated training data. To optimize the agent's iterative refinement capability, we also introduce Turn-Level GRPO (TL-GRPO), a lightweight, tool-integrated reinforcement learning algorithm that performs turn-level group sampling for fine-grained policy updates. Evaluated on 12 professional analog circuit tasks, SAAC trained with TL-GRPO achieves the best overall performance among the evaluated methods, outperforming Bayesian Optimization and other RL baselines. Moreover, SAAC demonstrates generalization to unseen circuit specifications, providing a practical foundation for LLM-driven automation in electronic design.

1 INTRODUCTION

Analog circuit sizing (ACS) is a critical task in electronic design automation (EDA), where engineers iteratively tune transistor parameters to meet multi-objective specifications. This process demands deep domain expertise, repeated expensive simulations, and careful balancing of conflicting objectives. Automating ACS with learning-based methods could significantly accelerate circuit design cycles, but existing approaches face fundamental limitations in this setting.

Prior machine learning methods for ACS have largely treated it as a black-box optimization problem, employing techniques like Bayesian optimization (Hu et al., 2018; Lyu et al., 2018). However, these methods ignore the reasoning process and prior circuit knowledge, rely on extensive sampling, and lack generalization across different circuit topologies. Recent work has explored LLM-based agents for circuit design (Yin et al., 2024; Ahmadzadeh et al., 2025; Liu & Chitnis, 2025; Chen et al., 2024; Wu et al., 2025) that leverage natural language reasoning to propose parameter adjustments. Yet these approaches either depend heavily on the base model's inherent capabilities without task-specific learning, or require costly human-annotated trajectories for supervised fine-tuning.

Reinforcement learning (Schulman et al., 2017; Ouyang et al., 2022) offers a promising direction for training LLMs to learn from environment feedback. However, directly applying standard RL algorithms with trajectory-level rewards (Shao et al., 2024; Jin et al., 2025; Li et al., 2025d) provides only coarse credit assignment in ACS.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

As illustrated in Figure 1, ACS constitutes an iterative optimization task where the agent interacts with a fixed environment state whose intrinsic properties remain unchanged across turns. Each action yields an observation and a state-independent reward, and the value of a trajectory is determined by the **best** reward achieved in any single turn. Consequently, effective optimization requires fine-grained credit assignment that distinguishes the quality of individual design proposals at each turn. Applying trajectory-level RL to this setting applies a uniform advantage signal across all turns in a trajectory, which yields noisy policy updates and suboptimal convergence.

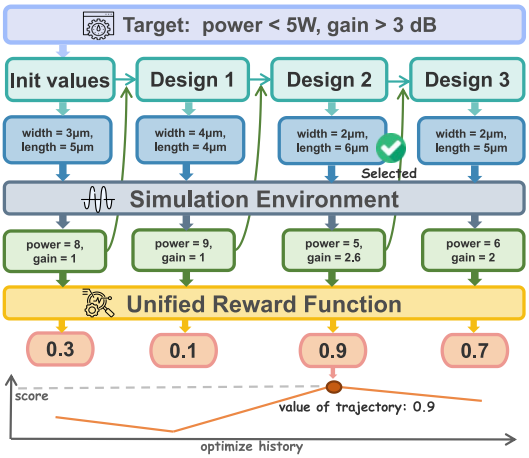


Figure 1: Illustration of analog circuit sizing as iterative optimization. The value of the entire trajectory is determined by the **best** design achieving the highest reward score among all turns, rather than cumulative returns.

To address these challenges, we formulate ACS as a reasoning-guided **iterative optimization** task and introduce SAAC (**S**izing **A**gent for **A**nalog **C**ircuit), an agentic framework tailored for automated circuit design. At the core of SAAC is Turn-Level GRPO (TL-GRPO), a lightweight agentic RL algorithm to adapt GRPO (Shao et al., 2024) for iterative optimization. Specifically, TL-GRPO begins by sampling a single full trajectory until interaction completion. Then, for each tool-call turn, it splits the trajectory at the turn's start and performs turn-level group sampling while preserving the interaction history as context. Due to the invariant environment state in ACS, a unified verifiable reward function can consistently evaluate outcomes across all turns, enabling fine-grained preference learning without introducing additional sampling cost compared to standard GRPO. Furthermore, to enable scalable training without human annotation, we develop an automatic data synthesis pipeline that randomizes circuit initial values and target specifications to generate diverse queries. Through SAAC's simulation environment and turn-level reward signals, the LLM learns directly from tool feedback to acquire professional analog-circuit reasoning and optimization capabilities.

We adopt Qwen3-30B-Instruct (Yang et al., 2025) as the base model and train SAAC using TL-GRPO. We evaluate it on 12 professional analog circuit tasks spanning amplifiers, data converters, and voltage regulators, leveraging a Cadence-based simulation environment. SAAC trained with TL-GRPO achieves the best overall performance among the evaluated methods on both in-domain and out-of-domain benchmarks, outperforming Bayesian optimization, standard GRPO, other multi-turn RL methods for iterative optimization (Baronio et al., 2025), and leading open-source LLMs. Related work is discussed in Appendix A.

In summary, our contributions are:

1. We introduce **SAAC**, an agentic framework for reasoning-guided analog circuit sizing that integrates professional-grade simulation environments with an automatic data synthesis pipeline, enabling scalable RL training without human annotation.

2. We formalize ACS as a reasoning-guided iterative optimization task and propose **TL-GRPO**, a turn-level RL algorithm that enables fine-grained preference optimization through history-conditioned group sampling and a unified verifiable reward.

3. Our 30B model trained with TL-GRPO on ACS tasks achieves the best overall performance among the evaluated methods and demonstrates generalization to unseen circuit topologies and process nodes, providing a foundation for future LLM applications in EDA.

2 PRELIMINARIES

2.1 PROBLEM FORMULATION FOR ITERATIVE OPTIMIZATION TASKS

Multi-turn tool-integrated reasoning is commonly modeled as a Partially Observable Markov Decision Process (POMDP) (Zhang et al., 2025a), defined by:

$$\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{O} \rangle, \tag{1}$$

where the agent receives an observation $o_t = O(s_t)$ based on the state $s_t \in \mathcal{S}$, takes an action $a_t \in \mathcal{A}_{\text{text}} \cup \mathcal{A}_{\text{tool_call}}$, and the environment transitions according to $\mathcal{P}(s_{t+1} \mid s_t, a_t)$. The objective of POMDP is to maximize the discounted cumulative reward:

$$\mathcal{J}_{\text{POMDP}}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T-1} \gamma^t R(s_t, a_t) \right], \quad 0 < \gamma < 1. \tag{2}$$

However, **iterative optimization** tasks, such as analog circuit sizing (ACS), constitute a distinct class where the agent aims to find the best possible solution within a fixed budget of turns. Formally, such tasks can be modeled as a **single-state POMDP**. Similar to multi-armed bandits, the agent interacts with a fixed environment that represents the evaluator of the optimization task. The state s, corresponding to the evaluator's intrinsic properties, remains unchanged throughout the interaction. This leads to a trivial state transition:

$$\mathcal{P}(s' \mid s, a) = \delta(s' = s), \quad \forall a \in \mathcal{A}, \tag{3}$$

where $\delta(\cdot)$ is the Dirac delta function. At each turn t, the agent proposes a candidate design $\mathbf{x}_t$ (via action a_t) and receives an observation $o_{t+1} = O(s, a_t)$ (e.g., performance metrics) and a reward $r_t = R(a_t)$ that depends only on the action. The underlying optimization objective is to find:

$$\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{X}} R(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d, \tag{4}$$

where $R : \mathcal{X} \to \mathbb{R}$ balances multiple objectives and $\mathcal{X} \subseteq \mathbb{R}^d$ is the d-dimensional design space.

Two fundamental properties distinguish iterative optimization from standard POMDP formulations:

Single-State Structure with Action-Dependent Feedback. Due to the invariant environment state (i.e., the fixed circuit topology and evaluation setup), observations and rewards vary solely based on the action a_t taken. The agent can utilize the entire history of interactions $h_t = (q, o_0, a_0, o_1, \ldots, a_{t-1}, o_t)$ to inform its decisions, enabling it to reason about the design space and improve proposals over successive turns. Crucially, the same deterministic reward function R uniformly evaluates every turn (e.g., Figure of Merit in ACS), making each reward $r_t = R(a_t)$ a self-contained measure of optimization quality.

Non-Cumulative, Maximum-Reward Objective. The goal is not to maximize the sum of rewards, but to achieve the highest possible reward in any single turn. Therefore, for iterative optimization, the value of a trajectory $\tau = (a_0, o_1, r_0, \ldots, a_{T-1}, o_T, r_{T-1})$ is:

$$G(\tau) = \max_{t \in [0, T-1]} r_t = \max_{t \in [0, T-1]} R(a_t). \tag{5}$$

Consequently, the optimization objective for an iterative optimization agent is:

$$\mathcal{J}_{\text{Opt}}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\max_{t \in [0, T-1]} R(a_t) \right]. \tag{6}$$

This objective creates a trade-off for policy optimization. Using $G(\tau)$ as a trajectory-level outcome directly reflects Eq. 6, but assigning its group-relative advantage uniformly to every turn does not distinguish the quality of individual proposals. TL-GRPO instead uses history-conditioned, turn-level reward comparisons as a dense surrogate, retaining the max-over-turn metric as the final objective while providing fine-grained learning signals for individual turns. This formulation captures ACS, where the circuit topology and simulation configuration remain fixed across turns while the agent uses accumulated feedback to improve successive proposals.

2.2 Group Relative Policy Optimization

Group Relative Policy Optimization (Shao et al., 2024) is a reinforcement learning method widely adopted to enhance the reasoning capabilities of LLMs. For each input query x, GRPO samples a group of G responses $\{y_1, y_2, \ldots, y_G\}$ from old policy $\pi_{\theta_{\text{old}}}$ and optimizes the following objective:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x,\{y_i\}_{i=1}^{G}\sim\pi_{\theta_{\text{old}}}} \Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|} \min\Big(w_{i,t}(\theta)A_{i,t}, \text{clip}\big(w_{i,t}(\theta), 1-\varepsilon, 1+\varepsilon\big)A_{i,t}\Big)\Bigg], \tag{7}$$

where $w_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t}|x, y_{i,<t})}{\pi_{\theta_{\text{old}}}(y_{i,t}|x, y_{i,<t})}$ is the token-level importance sampling ratio, ε is the clipping parameter. We also remove the KL divergence term following (Yu et al., 2025).

Given trajectory-level rewards $\{R_i^{\text{traj}}\}_{i=1}^{G}$, GRPO computes one normalized advantage per response:

$$A_{i,t} = A_i = \frac{R_i^{\text{traj}} - \text{mean}(\{R_i^{\text{traj}}\}_{i=1}^{G})}{\text{std}(\{R_i^{\text{traj}}\}_{i=1}^{G}) + \epsilon_A}, \quad \epsilon_A = 10^{-6}. \tag{8}$$

As shown in Eq. 8, the advantage $A_{i,t}$ is identical for all tokens within the same trajectory, applying a uniform, trajectory-level credit assignment. This design is effective in standard reasoning tasks, as test-time-scaling is shown to be beneficial for reasoning (Jaech et al., 2024). However, even when $R_i^{\text{traj}} = G(\tau_i)$ directly reflects the max-over-turn objective in Eq. 6, the resulting advantage provides no distinction between high- and low-quality turns. Consequently, applying vanilla GRPO to iterative optimization tasks like ACS often results in noisy policy updates and suboptimal convergence, as the uniform trajectory-level signal fails to guide turn-by-turn parameter refinement.

3 Method

To address the unique challenges of iterative optimization in analog circuit sizing, we introduce **SAAC** (**S**izing **A**gent for **A**nalog **C**ircuit), an agentic framework tailored for reasoning-guided circuit optimization. The framework integrates a distributed simulation environment (Sec. 3.2) with an automated data synthesis pipeline (Sec. 3.3). At its core, SAAC employs Turn-Level GRPO (TL-GRPO) (Sec. 3.1), a lightweight tool-integrated RL algorithm that performs history-conditioned group sampling to achieve fine-grained, turn-level preference optimization with verifiable turn-level reward (Sec. 3.1.2). As illustrated in Figure 2, the framework orchestrates a closed-loop interaction where the LLM agent proposes parameter updates, queries the fixed simulator via structured tool calls, and receives turn-level reward signals for policy updates.

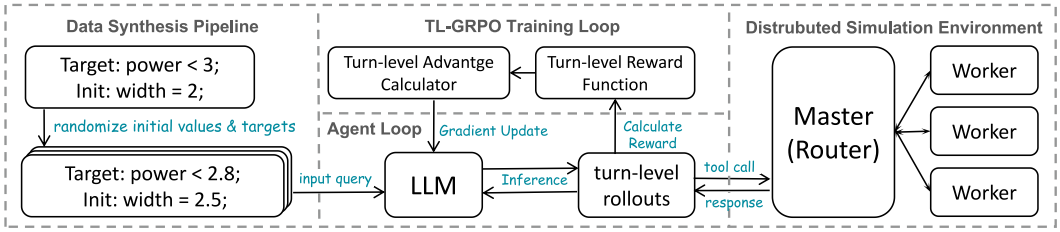


Figure 2: Overview of the SAAC framework. Our system integrates an automated data synthesis pipeline, a distributed master-worker simulation environment, and the TL-GRPO training loop.

3.1 TL-GRPO

Within the SAAC framework, TL-GRPO serves as the core optimization engine, designed to improve the max-reward objective through a fine-grained, history-conditioned surrogate. As illustrated

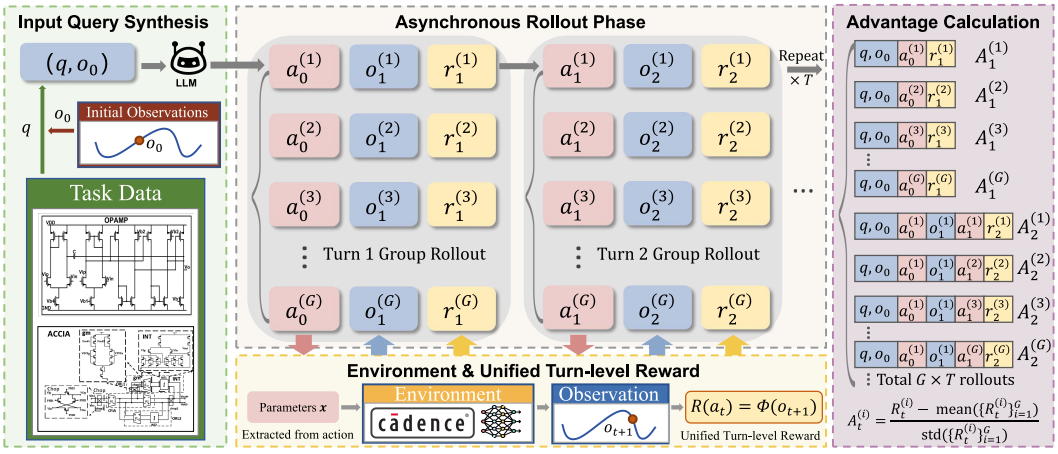


Figure 3: Overview of TL-GRPO. For an input query with initial observations, the agent asynchronously performs turn-level group sampling. Each sampled action $a_t^{(i)} \in \mathcal{A}_{\text{text}} \cup \mathcal{A}_{\text{tool_call}}$ receives an observation with a turn-level reward, and a group-relative advantage is subsequently estimated. All turn-level rollouts are then collected into one batch for policy gradient updates.

in Figure 3, our approach centers on two key strategies: (i) turn-level group sampling and (ii) unified verifiable turn rewards. With these modifications, TL-GRPO achieves fine-grained, turn-level preference optimization without introducing additional sampling cost compared to vanilla GRPO.

3.1.1 Turn-level Group Sampling

To enable turn-level value estimation without introducing additional sampling overhead, we depart from the trajectory-level group rollout strategy used in vanilla GRPO. In TL-GRPO, for each input query q, we begin by sampling **only one full trajectory** from the old policy $\pi_{\theta_{\text{old}}}$, denoted as $\tau = (o_0, a_0, o_1, a_1, ..., o_{T-1}, a_{T-1}, o_T)$.

We then split this trajectory at each turn, obtaining T distinct history contexts:

$$\begin{aligned} h_0 &= (q, o_0), \\ h_1 &= (q, o_0, a_0, o_1), \\ h_2 &= (q, o_0, a_0, o_1, a_1, o_2), \\ &\vdots \\ h_{T-1} &= (q, o_0, a_0, o_1, a_1, ..., o_{T-1}). \end{aligned} \tag{9}$$

For each history h_t, we sample $G-1$ additional actions from the rollout policy:

$$\begin{aligned} \{a_t^{(i)}\}_{i=1}^{G-1} &\sim \pi_{\theta_{\text{old}}}(a_t^{(i)} \mid h_t), \\ \text{where} \quad & a_t \in \mathcal{A}_{\text{text}} \cup \mathcal{A}_{\text{tool_call}}. \end{aligned} \tag{10}$$

This allows us to compute a group-relative advantage for each turn using the same normalized advantage estimation as in vanilla GRPO, but applied at turn level.

Inference Budget A key advantage of this design is that TL-GRPO maintains theoretically the same sampling cost as vanilla GRPO. The history context h_t for each turn is directly extracted from the initial trajectory rather than generated through additional rollouts. Let the expected number of turns in a trajectory be T and the group size be G. TL-GRPO therefore requires $T+(G-1)T = GT$ turn-level rollouts per query, matching the GT rollouts required by vanilla GRPO. This design avoids the substantial per-turn sampling overhead concerns noted in prior work (Feng et al., 2025b; Zeng et al., 2025), as TL-GRPO does not expand every sampled branch at each turn and roll it out to termination, but samples single-turn alternatives only from the single initial trajectory.

3.1.2 UNIFIED VERIFIABLE TURN-LEVEL REWARD FUNCTION

The second core component of TL-GRPO is the use of a unified, verifiable reward function for all turns. As described in Sec. 3.1.1, each action a_t results in an observation o_{t+1}, which comprises a vector of multiple performance metrics. The key step is to map this multi-objective vector into a scalar score that quantifies the overall quality of the proposed parameters a_t:

$$R(a_t) = \Phi(o_{t+1}), \quad \Phi : \mathbb{R}^{|o_{t+1}|} \to \mathbb{R} \tag{11}$$

where Φ is a deterministic aggregation function tailored to the ACS setting. Specifically, each circuit simulation yields a vector of performance metrics (e.g., gain, bandwidth, power), which we normalize and combine using a geometric mean. This design inherently penalizes any severely underperforming specification, encouraging the agent to pursue balanced improvements across all competing objectives rather than over-optimizing a single metric. A lightweight format penalty is further incorporated during training to enforce structured reasoning and tool-calling behavior. This unified reward provides a direct, verifiable signal for comparing alternative proposals under the same interaction history. TL-GRPO uses it as a dense turn-level preference signal rather than an exact long-horizon action-value estimate. Appendix C provides the complete reward formulation.

3.1.3 TRAINING OBJECTIVES

In this setting, the learning objective of TL-GRPO can be conceptualized as:

$$\begin{aligned} \mathcal{J}_{\text{TL-GRPO}}(\theta) = \mathbb{E}_{q,\tau\sim\pi_{\theta_{\text{old}}}} \Bigg[& \frac{1}{GT} \sum_{t=0}^{T-1} \sum_{i=1}^{G} \\ & \min \left(\frac{\pi_\theta(a_t^{(i)} \mid h_t)}{\pi_{\theta_{\text{old}}}(a_t^{(i)} \mid h_t)} A_t^{(i)}, \right. \\ & \left. \text{clip}\left(\frac{\pi_\theta(a_t^{(i)} \mid h_t)}{\pi_{\theta_{\text{old}}}(a_t^{(i)} \mid h_t)}, 1-\varepsilon, 1+\varepsilon \right) A_t^{(i)} \right) \Bigg], \end{aligned} \tag{12}$$

where each group of turn-level sampling outputs $\{a_t^{(i)}\}_{i=1}^{G} = \{a_t^{(1)}, a_t^{(2)}, \ldots, a_t^{(G)}\}$ is sampled from the old policy $\pi_{\theta_{old}}$ from every history context h_t. And turn-level group relative advantage $A_t^{(i)}$ is estimated by:

$$A_t^{(i)} = \frac{R_t^{(i)} - \text{mean}(\{R_t^{(i)}\}_{i=1}^{G})}{\text{std}(\{R_t^{(i)}\}_{i=1}^{G}) + \epsilon_A}. \tag{13}$$

Relation to the Max-over-Turn Objective. TL-GRPO uses a tractable, history-conditioned surrogate rather than an exact decomposition or unbiased estimator of Eq. 6. Consider format-compliant simulator proposals under a fixed history h_t; for these proposals, the auxiliary format penalty is zero, so the training reward equals the performance score $R(a)$. For $t > 0$, let $b_t = \max_{0\le j<t} R(a_j)$ denote the best previously observed reward. The immediate improvement from a candidate action is $\Delta_t(a \mid h_t) = [R(a) - b_t]_+$. Because all alternatives in a group share the same h_t, $R(a) > R(a')$ implies $\Delta_t(a \mid h_t) \ge \Delta_t(a' \mid h_t)$; at the first turn, candidates are ranked directly by R. Thus, among such proposals, the reward ordering used in Eq. 13 is consistent with immediate best-so-far improvement, while providing a denser preference when multiple candidates do not surpass b_t. This surrogate does not capture the delayed informational value of an action; we evaluate its effect on full-trajectory max reward empirically and discuss this limitation in Appendix I.

The algorithm of TL-GRPO is demonstrated in Appendix G, which provides fine-grained, turn-level preference optimization for multi-turn iterative optimization. Since $a_t \in \mathcal{A}_{\text{text}} \cup \mathcal{A}_{\text{tool_call}}$, this method offers turn-level supervision for both the reasoning process and the tool-calling strategy. Note that although the turn-level reward function is unified across turns, the advantage estimate $A_t^{(i)}$ is computed conditioned on the specific history context h_t. Consequently, by optimizing the conditional policy $\pi_\theta(a_t^{(i)} \mid h_t)$, the agent learns from historical interaction experience, which goes beyond merely fitting a static response distribution and enables the policy to leverage past context for learning from experience (Silver & Sutton, 2025).

3.2 SIMULATION ENVIRONMENT

We construct the SAAC simulation environment based on InternBootcamp (Li et al., 2025b). To ensure robust evaluation, we curate 12 real-world analog circuit sizing tasks spanning amplifiers, data converters, and voltage regulators, each implemented across distinct industrial Process Design Kits (PDKs) and netlist topologies. Each task defines a fixed set of tunable transistor parameters and multi-objective performance specifications (e.g., gain, bandwidth, power) that reflect production-grade design constraints. The agent's objective is to iteratively adjust these parameters to maximize a unified reward function derived from simulation feedback. Detailed circuit descriptions, optimization objectives, and task statistics are provided in Appendix B.

Circuit simulation in this domain presents two primary challenges: (1) complex environment configuration (e.g., Cadence Spectre) and (2) high computational latency per evaluation. To enable efficient multi-turn RL training, we implement a distributed master-worker architecture that decouples LLM inference from simulation execution, as illustrated in Figure 2. The agent operates within the training loop, issuing tool calls to a central master node that asynchronously schedules and executes simulations across independent workers. This architecture isolates the agent inference loop from the heavy simulation environment, ensuring scalable and stable environment interactions.

3.3 DATA SYNTHESIS PIPELINE

Unlike other ACS methods using supervised fine-tuning that depends on costly human-annotated trajectories, SAAC adopts an agentic reinforcement learning framework where the policy is optimized directly through tool-execution feedback. However, the training process still requires a diverse and scalable query distribution to ensure thorough exploration of the design space. To construct such a dataset within a tractable framework, we first define physically reasonable search boundaries for each tunable parameter (e.g., $width \in [0.4, 2]\,\mu\text{m}$) and limit the maximum interaction turns to $T = 5$. This constraint balances simulation cost and convergence likelihood.

Each analog circuit task initially corresponds to a single query with a fixed set of tunable parameters and production-realistic targets, which is insufficient for training. To create a diverse and scalable dataset from this simplified setup, we introduce two modifications under the guidance of domain experts: (1) **randomizing the initial values** of all tunable parameters within the predefined reasonable range for each query, and (2) **applying controlled random offsets** to original target specification values, generating diverse optimization targets within practical design margins. Given that a single task can involve dozens of parameters and multiple objectives, these variations yield substantial diversity, enabling many unique training queries without manual annotation.

Moreover, our prompt requires the LLM agent to first explicitly articulate its reasoning process within `<Analyze> </Analyze>` tags before calling the simulation tool and to conclude the interaction with `<Design End>`. This structured format clearly separates reasoning process from tool execution, aligning with the reasoning-guided iterative optimization paradigm. It also allows the reward function to distinguish agent-initiated termination from external truncation. Appendix H provides example prompts and tool descriptions.

4 EXPERIMENTS

We evaluate SAAC comprehensively on our ACS tasks.

4.1 EXPERIMENTAL SETUP

The 12 circuit tasks are partitioned into training and evaluation sets. 8 tasks are used for training and in-domain evaluation, while 4 Op-Amp tasks are held out for out-of-domain evaluation. We synthesize 10K training queries and generate 100 distinct queries per task for model evaluation, resulting in 800 in-domain and 400 out-of-domain instances.

Baseline Choices. Recent LLM-based ACS methods predominantly focus on single, proprietary circuit tasks and are typically closed-source, without public training data or standardized evaluation protocols; direct comparison under a unified multi-task benchmark is therefore infeasible.

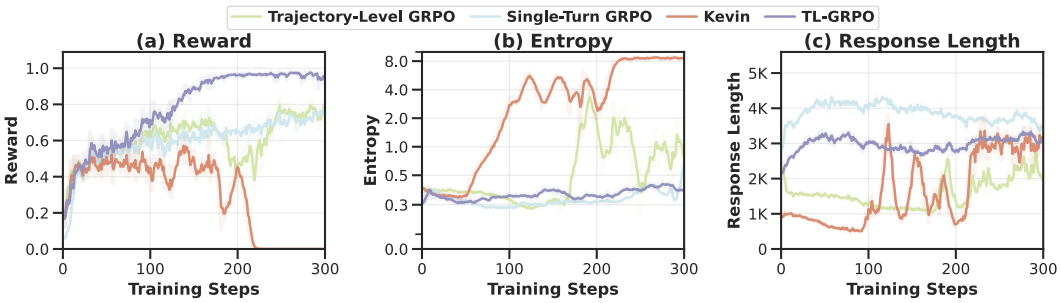


Figure 4: Comparison of training dynamics for four RL methods: Trajectory-Level GRPO, Single-Turn GRPO, Turn-Level GRPO, and Kevin. (a) Training reward score over steps. (b) Actor model generation entropy. (c) Average turn response length.

Table 1: In-domain evaluation results (8 trained tasks). *ST-iter* refers to the single-turn iterative evaluation protocol (Sec. 4.1).

Method	Eval	ACCIA	ADC1	ADC2	ADC3	ADC4	ADCL	OTA	SimLDO	**Avg.**
Bayesian Optimization		0.51	0.83	0.23	0.78	0.86	0.41	0.69	0.20	0.56
Qwen3-235B-Instruct-2507	Multi-turn	0.29	0.76	0.06	0.75	0.77	0.55	0.66	0.01	0.48
	ST-iter	0.33	0.88	0.07	0.85	0.87	0.60	0.66	0.01	0.53
DeepSeek-V3.2 (no-think)	Multi-turn	0.23	0.89	0.31	0.81	0.89	0.79	0.63	0.46	0.63
	ST-iter	0.24	0.93	0.34	0.83	0.88	0.60	0.67	0.50	0.62
DeepSeek-V3.2 (thinking)	Multi-turn	0.26	0.85	0.31	0.78	0.81	0.78	0.65	0.55	0.62
	ST-iter	0.25	0.89	0.33	0.80	0.74	0.53	0.66	0.65	0.61
Qwen3-30B-Instruct-2507	Multi-turn	0.20	0.65	0.15	0.71	0.76	0.50	0.59	0.01	0.45
	ST-iter	0.18	0.79	0.21	0.75	0.83	0.39	0.65	0.00	0.47
+ Trajectory-Level GRPO	Multi-turn	0.69	0.98	0.74	0.99	0.98	0.92	0.87	0.61	0.85
+ Single-Turn GRPO	ST-iter	0.68	0.99	0.61	0.93	0.97	0.93	0.93	0.57	0.83
+ Kevin	Multi-turn	0.29	0.86	0.34	0.84	0.83	0.50	0.59	0.13	0.55
+ Turn-Level GRPO (Ours)	Multi-turn	**0.94**	**1.0**	**0.91**	**1.0**	**1.0**	**0.98**	**0.99**	**0.97**	**0.97**

We therefore compare against Bayesian Optimization, strong open-source LLMs, and controlled RL baselines in the same interactive framework. Among multi-turn RL methods, GiGPO (Feng et al., 2025b) relies on anchor-state grouping across trajectories, which requires repeated intermediate decision states that are essentially absent in continuous ACS observations. We therefore include Kevin (Baronio et al., 2025), a multi-turn method for iterative optimization, together with Trajectory-Level and Single-Turn GRPO.

We compare four RL methods to isolate the impact of turn-level optimization: (1) TL-GRPO (multi-turn rollout, turn-level optimization), (2) Trajectory-Level GRPO (trajectory-level optimization using the max-over-turn trajectory reward, directly matching the task objective), (3) Single-Turn GRPO (single-turn optimization without history), and (4) Kevin (multi-turn optimization with advantages normalized across all turns in a rollout group). We use Qwen3-30B-A3B-Instruct (Yang et al., 2025) as the base model, with batch size 32, rollout group size $G = 8$, and at most $T = 5$ turns per query in multi-turn settings. Further training details are in Appendix D.

During evaluation, we compare these four RL methods, Bayesian Optimization (using the implementation of (Nogueira, 2014–)), and strong open-source LLMs (Qwen3-235B-Instruct and DeepSeek-V3.2)[1]. Each query starts from one random initial simulation point, permits at most 5 tool calls, and uses the best candidate score as final. We use two LLM evaluation protocols: (i) multi-turn evaluation for TL-GRPO, Trajectory-Level GRPO, and Kevin; and (ii) single-turn iterative evaluation, where each design uses only the original query and latest simulation result, matching the history-free context used during Single-Turn GRPO training.

Table 2: Out-of-domain evaluation results (4 unseen tasks). *ST-iter* refers to the single-turn iterative evaluation protocol (Sec. 4.1).

Method	Eval	OPAMP1	OPAMP2	OPAMP3	OPAMP4	**Avg.**
Bayesian Optimization		**0.36**	0.01	0.08	0.24	0.17
Qwen3-235B-Instruct-2507	Multi-turn	0.21	0.04	0.08	0.19	0.13
	ST-iter	0.23	0.17	0.15	0.24	0.20
DeepSeek-V3.2 (no-think)	Multi-turn	0.25	0.23	0.24	0.30	0.26
	ST-iter	0.22	0.26	0.24	0.24	0.24
DeepSeek-V3.2 (thinking)	Multi-turn	0.22	0.23	0.26	0.26	0.24
	ST-iter	0.20	0.28	**0.32**	0.23	0.26
Qwen3-30B-Instruct-2507	Multi-turn	0.14	0.01	0.05	0.16	0.09
	ST-iter	0.14	0.04	0.06	0.15	0.10
+ Trajectory-Level GRPO	Multi-turn	0.15	0.18	0.20	0.17	0.18
+ Single-Turn GRPO	ST-iter	0.20	0.15	0.26	0.25	0.21
+ Kevin	Multi-turn	0.11	0.18	0.28	0.09	0.16
+ Turn-Level GRPO (Ours)	Multi-turn	0.20	**0.32**	0.31	**0.58**	**0.35**

4.2 Main Results

Table 1 and 2 present the evaluation performance of TL-GRPO and all baseline methods under the same simulation budget (a maximum of 5 tool-call turns). All reported scores are averaged over three independent evaluation runs. We report results from the best checkpoint for each method. Appendix H provides example model outputs.

On in-domain evaluation set (Table 1), TL-GRPO achieves the highest scores across all 8 trained ACS tasks, with an average score of 0.97. This demonstrates that our method enables the 30B model to master the trained tasks effectively. Notably, although Trajectory-Level GRPO directly uses the max-over-turn trajectory score, TL-GRPO attains higher full-trajectory performance, empirically supporting the benefit of its finer-grained surrogate signal.

The out-of-domain (OOD) evaluation (Table 2) holds greater practical significance, as real-world optimization tasks like ACS are computationally expensive, limiting the feasibility of extensive sampling or iteration for new problems and making generalization important. Here, TL-GRPO again achieves the highest average max-over-turn score (0.35), surpassing Bayesian Optimization (0.17), Kevin (0.16), and powerful LLMs including Qwen3-235B-Instruct (0.20) and DeepSeek-V3.2 (0.26), suggesting that the benefit of its finer-grained surrogate also transfers to held-out tasks. Moreover, TL-GRPO also achieves the highest all-specification success rate among the matched Qwen3-30B training variants (see Appendix E for details). Figure 4 shows that TL-GRPO reaches the highest training reward while maintaining stable entropy and response length; further discussion and per-turn analysis are provided in Appendix F.1 and F.2.

5 Conclusion

In this work, we introduce SAAC, an agentic framework tailored for reasoning-guided analog circuit sizing. By integrating a professional simulation environment, an automated data synthesis pipeline, and Turn-Level GRPO (TL-GRPO) as its core optimization engine, SAAC enables scalable, annotation-free RL training. Across 12 professional ACS tasks, SAAC achieves the best overall performance among the evaluated Bayesian optimization, RL, and LLM baselines. These results show that history-conditioned turn-level supervision can improve the final max-over-turn outcome while maintaining stable training dynamics. Moreover, SAAC exhibits generalization to unseen circuit topologies and process nodes, providing a practical foundation for future LLM applications in electronic design automation.

[1]Due to data confidentiality demands from collaborators, we are unable to use APIs from closed-source models for evaluation.

AI USE STATEMENT

Generative AI tools assisted with manuscript editing, code development and debugging. All AI-assisted outputs were reviewed and verified by the authors, who take responsibility for the final content of this work. The use of LLMs as components of the proposed system and data-synthesis pipeline is described in the main text.

REPRODUCIBILITY STATEMENT

We provide the algorithm, reward definition, training and evaluation settings, task statistics, and representative data examples in the main paper and appendices. The circuit designs, process design kits, and professional simulation environment are proprietary, and this restriction is stated alongside the experimental details.

REFERENCES

Mohsen Ahmadzadeh, Kaichang Chen, and Georges G. E. Gielen. Anaflow: Agentic llm-based workflow for reasoning-driven explainable and sample-efficient analog circuit sizing. *CoRR*, abs/2511.03697, 2025. doi: 10.48550/ARXIV.2511.03697. URL https://doi.org/10.48550/arXiv.2511.03697.

Carlo Baronio, Pietro Marsella, Ben Pan, Simon Guo, and Silas Alberti. Kevin: Multi-turn rl for generating cuda kernels. *arXiv preprint arXiv:2507.11948*, 2025.

Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, Chengjun Xiao, Chengyu Du, Chi Zhang, Chu Qiao, Chunhao Zhang, Chunhui Du, Congchao Guo, Da Chen, Deming Ding, Dianjun Sun, Dong Li, Enwei Jiao, Haigang Zhou, Haimo Zhang, Han Ding, Haohai Sun, Haoyu Feng, Huaiguang Cai, Haichao Zhu, Jian Sun, Jiaqi Zhuang, Jiaren Cai, Jiayuan Song, Jin Zhu, Jingyang Li, Jinhao Tian, Jinli Liu, Junhao Xu, Junjie Yan, Junteng Liu, Junxian He, Kaiyi Feng, Ke Yang, Kecheng Xiao, Le Han, Leyang Wang, Lianfei Yu, Liheng Feng, Lin Li, Lin Zheng, Linge Du, Lingyu Yang, Lunbin Zeng, Minghui Yu, Mingliang Tao, Mingyuan Chi, Mozhi Zhang, Mujie Lin, Nan Hu, Nongyu Di, Peng Gao, Pengfei Li, Pengyu Zhao, Qibing Ren, Qidi Xu, Qile Li, Qin Wang, Rong Tian, Ruitao Leng, Shaoxiang Chen, Shaoyu Chen, Shengmin Shi, Shitong Weng, Shuchang Guan, Shuqi Yu, Sichen Li, Songquan Zhu, Tengfei Li, Tianchi Cai, Tianrun Liang, Weiyu Cheng, Weize Kong, Wenkai Li, Xiancai Chen, Xiangjun Song, Xiao Luo, Xiao Su, Xiaobo Li, Xiaodong Han, Xinzhu Hou, Xuan Lu, Xun Zou, Xuyang Shen, Yan Gong, Yan Ma, Yang Wang, Yiqi Shi, Yiran Zhong, and Yonghong Duan. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *CoRR*, abs/2506.13585, 2025a. doi: 10.48550/ARXIV.2506.13585. URL https://doi.org/10.48550/arXiv.2506.13585.

Jiangjie Chen, Qianyu He, Siyu Yuan, Aili Chen, Zhicheng Cai, Weinan Dai, Hongli Yu, Qiying Yu, Xuefeng Li, Jiaze Chen, Hao Zhou, and Mingxuan Wang. Enigmata: Scaling logical reasoning in large language models with synthetic verifiable puzzles. *CoRR*, abs/2505.19914, 2025b. doi: 10.48550/ARXIV.2505.19914. URL https://doi.org/10.48550/arXiv.2505.19914.

Zihao Chen, Jiangli Huang, Yiting Liu, Fan Yang, Li Shang, Dian Zhou, and Xuan Zeng. Artisan: Automated operational amplifier design via domain-specific large language model. In Vivek De (ed.), *Proceedings of the 61st ACM/IEEE Design Automation Conference, DAC 2024, San Francisco, CA, USA, June 23-27, 2024*, pp. 39:1–39:6. ACM, 2024. doi: 10.1145/3649329.3655903. URL https://doi.org/10.1145/3649329.3655903.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL https://doi.org/10.48550/arXiv.2501.12948.

Yifeng Ding, Hung Le, Songyang Han, Kangrui Ruan, Zhenghui Jin, Varun Kumar, Zijian Wang, and Anoop Deoras. Empowering multi-turn tool-integrated reasoning with group turn policy optimization. *arXiv preprint arXiv:2511.14846*, 2025.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *CoRR*, abs/2504.11536, 2025a. doi: 10.48550/ARXIV.2504.11536. URL https://doi.org/10.48550/arXiv.2504.11536.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. *CoRR*, abs/2505.10978, 2025b. doi: 10.48550/ARXIV.2505.10978. URL https://doi.org/10.48550/arXiv.2505.10978.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pp. 10835–10866. PMLR, 2023.

Hanbin Hu, Peng Li, and Jianhua Z. Huang. Parallelizable bayesian optimization for analog and mixed-signal rare failure detection with high coverage. In Iris Bahar (ed.), *Proceedings of the International Conference on Computer-Aided Design, ICCAD 2018, San Diego, CA, USA, November 05-08, 2018*, pp. 98. ACM, 2018. doi: 10.1145/3240765.3240835. URL https://doi.org/10.1145/3240765.3240835.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, and Ilge Akkaya. Openai o1 system card. *CoRR*, abs/2412.16720, 2024. doi: 10.48550/ARXIV.2412.16720. URL https://doi.org/10.48550/arXiv.2412.16720.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for LLM agent reinforcement learning. *CoRR*, abs/2509.21240, 2025. doi: 10.48550/ARXIV.2509.21240. URL https://doi.org/10.48550/arXiv.2509.21240.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516, 2025. doi: 10.48550/ARXIV.2503.09516. URL https://doi.org/10.48550/arXiv.2503.09516.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *CoRR*, abs/2406.18629, 2024. doi: 10.48550/ARXIV.2406.18629. URL https://doi.org/10.48550/arXiv.2406.18629.

Peiji Li, Kai Lv, Yunfan Shao, Yichuan Ma, Linyang Li, Xiaoqing Zheng, Xipeng Qiu, and Qipeng Guo. Fastmcts: A simple sampling strategy for data synthesis. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 24405–24422. Association for Computational Linguistics, 2025a. URL https://aclanthology.org/2025.acl-long.1190/.

Peiji Li, Jiasheng Ye, Yongkang Chen, Yichuan Ma, Zijie Yu, Kedi Chen, Ganqu Cui, Haozhan Li, Jiacheng Chen, Chengqi Lyu, Wenwei Zhang, Linyang Li, Qipeng Guo, Dahua Lin, Bowen Zhou, and Kai Chen. Internbootcamp technical report: Boosting LLM reasoning with verifiable

task scaling. *CoRR*, abs/2508.08636, 2025b. doi: 10.48550/ARXIV.2508.08636. URL https://doi.org/10.48550/arXiv.2508.08636.

Xiaozhe Li, Xinyu Fang, Shengyuan Ding, Linyang Li, Haodong Duan, Qingwen Liu, and Kai Chen. Np-engine: Empowering optimization reasoning in large language models with verifiable synthetic np problems, 2025c. URL https://arxiv.org/abs/2510.16476.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated RL. *CoRR*, abs/2503.23383, 2025d. doi: 10.48550/ARXIV.2503.23383. URL https://doi.org/10.48550/arXiv.2503.23383.

Chang Liu and Danial Chitnis. Eesizer: Llm-based AI agent for sizing of analog and mixed signal circuit. *CoRR*, abs/2509.25510, 2025. doi: 10.48550/ARXIV.2509.25510. URL https://doi.org/10.48550/arXiv.2509.25510.

Junteng Liu, Yuanxiang Fan, Zhuo Jiang, Han Ding, Yongyi Hu, Chi Zhang, Yiqi Shi, Shitong Weng, Aili Chen, Shiqi Chen, Yunan Huang, Mozhi Zhang, Pengyu Zhao, Junjie Yan, and Junxian He. Synlogic: Synthesizing verifiable reasoning data at scale for learning logical reasoning and beyond. *CoRR*, abs/2505.19641, 2025. doi: 10.48550/ARXIV.2505.19641. URL https://doi.org/10.48550/arXiv.2505.19641.

Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, and Xuan Zeng. Batch bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design. In *International conference on machine learning*, pp. 3306–3314. PMLR, 2018.

Yichuan Ma, Linyang Li, Yongkang Chen, Peiji Li, Jiasheng Ye, Qipeng Guo, Dahua Lin, and Kai Chen. Mixing expert knowledge: Bring human thoughts back to the game of go. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=baNBqpzvMT.

Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014–. URL https://github.com/bayesian-optimization/BayesianOptimization.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *CoRR*, abs/2504.13958, 2025. doi: 10.48550/ARXIV.2504.13958. URL https://doi.org/10.48550/arXiv.2504.13958.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

David Silver and Richard S Sutton. Welcome to the era of experience. *Google AI*, 1, 2025.

Yi Su, Dian Yu, Linfeng Song, Juntao Li, Haitao Mi, Zhaopeng Tu, Min Zhang, and Dong Yu. Crossing the reward bridge: Expanding RL with verifiable rewards across diverse domains. *CoRR*, abs/2503.23829, 2025. doi: 10.48550/ARXIV.2503.23829. URL https://doi.org/10.48550/arXiv.2503.23829.

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. SPA-RL: reinforcing LLM agents via stepwise progress attribution. *CoRR*, abs/2505.20732, 2025a. doi: 10.48550/ARXIV.2505.20732. URL https://doi.org/10.48550/arXiv.2505.20732.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. OTC: optimal tool calls via reinforcement learning. *CoRR*, abs/2504.14870, 2025b. doi: 10.48550/ARXIV.2504.14870. URL https://doi.org/10.48550/arXiv.2504.14870.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Yifan Wei, Xiaoyan Yu, Yixuan Weng, Tengfei Pan, Angsheng Li, and Li Du. Autotir: Autonomous tools integrated reasoning via reinforcement learning. *CoRR*, abs/2507.21836, 2025. doi: 10.48550/ARXIV.2507.21836. URL https://doi.org/10.48550/arXiv.2507.21836.

Xinyue Wu, Fan Hu, Jani Babu Shaik, Yi Zhao, and Xinfei Guo. Easysize: Elastic analog circuit sizing via llm-guided heuristic search. *CoRR*, abs/2508.05113, 2025. doi: 10.48550/ARXIV.2508.05113. URL https://doi.org/10.48550/arXiv.2508.05113.

Yuxi Xie, Anirudh Goyal, Wenyue Zheng, Min-Yen Kan, Timothy P. Lillicrap, Kenji Kawaguchi, and Michael Shieh. Monte carlo tree search boosts reasoning via iterative preference learning. *CoRR*, abs/2405.00451, 2024. doi: 10.48550/ARXIV.2405.00451. URL https://doi.org/10.48550/arXiv.2405.00451.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jian Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Yuxuan Yin, Yu Wang, Boxun Xu, and Peng Li. ADO-LLM: analog design bayesian optimization with in-context learning of large language models. In Jinjun Xiong and Robert Wille (eds.), *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2024, Newark Liberty International Airport Marriott, NJ, USA, October 27-31, 2024*, pp. 81:1–81:9. ACM, 2024. doi: 10.1145/3676536.3676816. URL https://doi.org/10.1145/3676536.3676816.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Weinan Dai, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. DAPO: an open-source LLM reinforcement learning system at scale. *CoRR*, abs/2503.14476, 2025. doi: 10.48550/ARXIV.2503.14476. URL https://doi.org/10.48550/arXiv.2503.14476.

Siliang Zeng, Quan Wei, William Brown, Oana Frunza, Yuriy Nevmyvaka, and Mingyi Hong. Reinforcing multi-turn reasoning in LLM agents via turn-level credit assignment. *CoRR*, abs/2505.11821, 2025. doi: 10.48550/ARXIV.2505.11821. URL https://doi.org/10.48550/arXiv.2505.11821.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Yue Liao, Hongru Wang, Mengyue Yang, Heng Ji, Michael Littman, Jun Wang, Shuicheng Yan, Philip Torr, and Lei Bai. The landscape of agentic reinforcement learning for llms: A survey. *CoRR*, abs/2509.02547, 2025a. doi: 10.48550/ARXIV.2509.02547. URL https://doi.org/10.48550/arXiv.2509.02547.

Jingyi Zhang, Jiaxing Huang, Huanjin Yao, Shunyu Liu, Xikun Zhang, Shijian Lu, and Dacheng Tao. R1-VL: learning to reason with multimodal large language models via step-wise group relative policy optimization. *CoRR*, abs/2503.12937, 2025b. doi: 10.48550/ARXIV.2503.12937. URL https://doi.org/10.48550/arXiv.2503.12937.

Ruiyang Zhang, Dongzhan Zhou, and Zhedong Zheng. Sketchthinker-r1: Towards efficient sketch-style reasoning in large multimodal models, 2026. URL https://arxiv.org/abs/2601.02825.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

A Related Work

A.1 Analog Circuit Sizing

Analog circuit sizing (ACS) is a classical optimization task in which domain experts tune circuit parameters to meet specifications such as gain and power. Prior machine learning work largely treats ACS as black-box optimization, using Bayesian optimization (Hu et al., 2018; Lyu et al., 2018), with limited generalization across circuit topologies. Recent LLM approaches include multi-agent frameworks (Yin et al., 2024; Ahmadzadeh et al., 2025; Liu & Chitnis, 2025; Chen et al., 2024) and supervised fine-tuning (Wu et al., 2025), but rely on base-model capabilities or human-annotated data. SAAC instead learns from simulator feedback through turn-level RL.

A.2 RL for LLM Reasoning

RL methods such as PPO (Schulman et al., 2017), DPO (Rafailov et al., 2023), GRPO (Shao et al., 2024), DAPO (Yu et al., 2025), GSPO (Zheng et al., 2025), and CISPO (Chen et al., 2025a) perform sequence-level preference optimization, and outcome-based GRPO underpins much recent RLVR work (Liu et al., 2025; Chen et al., 2025b; Su et al., 2025; Ma et al., 2025; Zhang et al., 2026; Li et al., 2025c). Since LLMs often reason with Chain-of-Thought (Wei et al., 2022), step-level supervision (Lai et al., 2024; Xie et al., 2024; Li et al., 2025a; Shao et al., 2024; Zhang et al., 2025b) can provide finer signals, but process reward models require training or heuristics and risk reward hacking (DeepSeek-AI, 2025; Gao et al., 2023), while Monte Carlo value estimates add inference cost. Thus, outcome-based GRPO remains the dominant RLVR paradigm.

A.3 RL for Tool-Integrated Reasoning

Tool-integrated RL methods (Li et al., 2025d; Qian et al., 2025; Wang et al., 2025b; Feng et al., 2025a; Wei et al., 2025; Zhang et al., 2025b) typically optimize trajectory-level rewards, obscuring the contribution of individual tool calls. Recent methods provide finer signals: SPARL (Wang et al., 2025a), MT-PPO (Zeng et al., 2025), and GTPO (Ding et al., 2025) use heuristic intermediate rewards; GiGPO (Feng et al., 2025b) groups turns by anchor state; and Tree-GRPO (Ji et al., 2025) back-propagates trajectory rewards as process supervision. These approaches target general POMDPs or aggregated states. For iterative optimization, Kevin (Baronio et al., 2025) assigns discounted returns over current and future scores but normalizes them across all turn samples in a rollout group without conditioning the comparison group on interaction history. TL-GRPO instead groups actions under the same history, yielding history-aware turn-level advantages.

B Details of ACS Circuits

Our data comprises **12** practical analog circuit sizing tasks, implemented across multiple industrial process technologies. All tasks maintain a static netlist topology throughout the optimization process, with only transistor parameter values changing across turns, which strictly satisfies the invariant-state assumption of our framework. The tasks exhibit significant diversity in both circuit complexity and optimization scale (see Table 3 for detailed statistics).

- **ACCIA:** AC-coupled instrumentation amplifier (0.18-µm process).
- **OPAMP1–4:** Four operational amplifiers utilizing different process nodes: TSMC 40nm Mixed-Signal RF, TSMC 0.18µm RF, TSMC 0.18µm Mixed-Signal, and SMIC 40nm Logic.
- **OTA:** Telescopic-cascode operational transconductance amplifier (TSMC 0.18µm RF).
- **ADC1–ADCL:** A family of analog-to-digital converters with 8–10 bit resolution, operating at sampling frequencies from 20 MHz to 1 GHz. Process technologies include SMIC 0.18µm, TSMC 28nm, and TSMC 65nm.
- **SimLDO:** A low-dropout linear voltage regulator (LDO). This power-management circuit represents a key category of analog design focused on voltage regulation.

Table 3: Key characteristics of the analog circuit sizing tasks in our dataset. We report the number of tunable parameters, the number of optimization objectives (e.g., gain, bandwidth, power), and the token length of the circuit netlist. The tilde ($\sim$) denotes approximate values. OPAMP1–4 share similar netlist structures (with PDK variations), resulting in comparable token lengths and identical parameter counts.

Circuit	ACCIA	ADC1	ADC2	ADC3	ADC4	ADCL	OTA	SimLDO	OPAMP1–4
Tunable Parameters	38	28	24	24	35	17	10	37	19
Number of Objectives	4	4	4	4	4	4	3	12	4
Token Length	12,208	5,244	4,054	4,735	5,418	3,550	1,185	3,289	$\sim$800

C Implementation Details of Unified Turn-level Reward Function

As formalized in Section 3.1.2, TL-GRPO employs a unified verifiable reward function Φ that maps the observation o_{t+1} (the tool response resulting from action a_t) into a scalar turn-level reward $R_t = \Phi(o_{t+1})$. For the Analog Circuit Sizing task, the observation o_{t+1} contains the values of multiple performance metrics (e.g., gain, bandwidth, phase margin, power consumption) resulting from the proposed circuit parameters.

Let $\mathcal{M} = \{m_1, m_2, \ldots, m_M\}$ denote the set of M optimization objectives. Each objective m_j has a current value v_j (extracted from tool response o_{t+1}) and a target specification s_j that defines the desired performance. The specifications can be of three types: **lower-bound** (e.g., minimum gain), **upper-bound** (e.g., maximum power), and **range** (e.g., output voltage range). For each objective, we define a normalized score $p_j(v_j, s_j) \in [0, 1]$ that quantifies how well v_j meets s_j, using threshold parameters τ_j^ℓ and τ_j^u to create smooth transitions between acceptable and unacceptable performance. These thresholds are set proportionally to the specification values (e.g., $\tau_j^\ell = \alpha|s_j|, \tau_j^u = \beta|s_j|$) to maintain physical meaningfulness. The normalized score function $p_j(v_j, s_j)$ for each specification type is defined as follows:

1. Lower-bound specification (v_j should satisfy $v_j \geq s_j$):

$$p_j^{\text{lower}}(v_j, s_j) = \begin{cases} 0, & v_j < s_j - \tau_j^\ell \\ \left(\frac{v_j - (s_j - \tau_j^\ell)}{\tau_j^\ell}\right)^2, & s_j - \tau_j^\ell \leq v_j < s_j \\ 1, & v_j \geq s_j \end{cases} \tag{14}$$

2. Upper-bound specification (v_j should satisfy $v_j \leq s_j$):

$$p_j^{\text{upper}}(v_j, s_j) = \begin{cases} 1, & v_j \leq s_j \\ \left(\frac{s_j + \tau_j^u - v_j}{\tau_j^u}\right)^3, & s_j < v_j \leq s_j + \tau_j^u \\ 0, & v_j > s_j + \tau_j^u \end{cases} \tag{15}$$

3. Range specification (v_j should fall within an interval $s_j = [\ell_j, u_j]$):

$$p_j^{\text{range}}(v_j, s_j) = \begin{cases} 0, & v_j < \ell_j - \tau_j^\ell \\ \left(\frac{v_j - (\ell_j - \tau_j^\ell)}{\tau_j^\ell}\right)^2, & \ell_j - \tau_j^\ell \leq v_j < \ell_j \\ 1, & \ell_j \leq v_j \leq u_j \\ \left(\frac{u_j + \tau_j^u - v_j}{\tau_j^u}\right)^3, & u_j < v_j \leq u_j + \tau_j^u \\ 0, & v_j > u_j + \tau_j^u \end{cases} \tag{16}$$

These continuous, piecewise-defined functions—composed of constant, quadratic, and cubic segments—are constructed to remain within $[0, 1]$, ensuring that their geometric mean (Eq. 17) also falls within $[0, 1]$ and provides a normalized overall score. This design delivers smooth penalties near specification boundaries, encouraging the LLM agent to robustly meet targets rather than marginally

violating them. The overall performance reward $P_t \in [0, 1]$ is computed as the geometric mean of all per-objective scores:

$$P_t = \left(\prod_{j=1}^{M} p_j(v_j, s_j) \right)^{1/M} \tag{17}$$

The geometric mean ensures that the agent must balance all objectives simultaneously, as a low score in any single objective will significantly reduce P_t. This unified turn-level reward function serves like a Figure of Merit (FoM) in analog circuit design, but normalized to $[0, 1]$ for consistent evaluation across different tasks.

Additionally, we introduce a format penalty $F_t \in [-1, 0]$ during training that penalizes violations of the required interaction format, such as missing `<Analyze>` or `<Design End>` tags, or exceeding the maximum tool-calling turns T. In evaluation, for fair comparison, we do not apply format penalty. The final reward is:

$$\Phi_t = \begin{cases} \max\big(0,\ P_t + F_t\big), & \text{for training} \\ P_t, & \text{for eval} \end{cases} \tag{18}$$

where $\Phi_t \in [0, 1]$. This design provides a consistent, verifiable, and fine-grained turn-level reward signal that directly aligns with the multi-objective nature of the ACS task while encouraging proper interaction behavior.

During preliminary development, we empirically evaluated alternative aggregation functions, including arithmetic mean, weighted sum, and min-operator. The geometric mean consistently yielded more stable policy updates, with resulting scores that more accurately reflect true circuit performance. This design strictly penalizes severe violations in any single objective, aligning with standard Figure-of-Merit (FoM) practices in EDA. Furthermore, we also validated our reward formulation through consultations with domain experts in analog circuit sizing, who confirmed its generality and robustness across diverse design specifications.

D Training Setup Details

We provide a comprehensive summary of the hyperparameters and implementation details used for all RL training paradigms. All RL experiments use Qwen3-30B-A3B-Instruct as the base policy and are trained with the VeRL codebase (Sheng et al., 2024) using 32 NVIDIA H200 GPUs. We use a global prompt batch size of 32 and learning rate of 1×10^{-6} for 1 epoch. For multi-turn paradigms (TL-GRPO, Trajectory-Level GRPO, and Kevin), each query is allowed a maximum of $T = 5$ tool-call turns. During the rollout phase, the sampling group size G is 8 (applied at turn level for TL-GRPO and at sequence level for Trajectory-Level GRPO, Single-Turn GRPO, and Kevin) with a temperature of 1.0 and top-p sampling of 0.95. For Kevin, the discounted-return factor is $\gamma = 0.4$. To accommodate long reasoning traces and tool outputs, we set the maximum prompt length to 15,000 tokens (for long circuit netlist content) and the maximum response length per turn to 4,096 tokens. Following practices from Yu et al. (2025), we employ clip ratios $\varepsilon_{\text{low}} = 0.2$ and $\varepsilon_{\text{high}} = 0.28$, and we do not apply KL-divergence regularization.

E All-Specification Success Analysis

Beyond the geometric-mean score used in the main results, Table 4 compares all-specification success among the matched Qwen3-30B training variants in out-of-domain evaluation. TL-GRPO achieves the highest success rate (25.50%) among all matched training variants.

F Additional Experimental Analysis

Our analysis yields three key findings:

1. Compared to Bayesian Optimization, SAAC demonstrates the value of **reasoning-guided iterative optimization**, where prior knowledge and reasoning capabilities enable better generalization to unseen tasks.

Table 4: OOD all-specification success for matched Qwen3-30B training variants.

Method	Success (%)
Qwen3-30B-Instruct-2507, Multi-turn	8.50
Qwen3-30B-Instruct-2507, ST-iter	7.25
+ Trajectory-Level GRPO	5.25
+ Single-Turn GRPO	21.25
+ Kevin	5.50
+ Turn-Level GRPO (Ours)	**25.50**

2. Compared to Trajectory-Level GRPO, TL-GRPO's superior performance validates that **turn-level** fine-grained optimization strengthens reasoning capability more effectively than trajectory-level, outcome-based rewards.

3. Compared to Single-Turn GRPO and Kevin, TL-GRPO demonstrates the benefit of history-conditioned group sampling: comparing turn-level actions under the same interaction history enables the policy to *learn from experience* across iterations.

F.1 Analysis of Training Dynamics

Figure 4 compares the training reward, generation entropy, and average response length across TL-GRPO, Trajectory-Level GRPO, Single-Turn GRPO, and Kevin, effectively serving as an analysis that isolates the impact of turn-level optimization in our method relative to the baselines.

The reward curve in Figure 4(a) demonstrates that TL-GRPO achieves faster convergence and reaches higher final rewards than the baselines. Kevin improves during early training but becomes unstable after 200 steps. Furthermore, as shown in Figure 4(c) and 4(b), TL-GRPO maintains stable response lengths and well-behaved generation entropy throughout training, despite the challenge of optimizing sequences that can potentially reach 35k tokens (due to a maximum of 15k-token prompt and up to 5 $\times$ 4k-token turn response). The entropy does not collapse to low values—which would indicate reduced exploration—nor does it exhibit sharp fluctuations, suggesting stable and controlled policy updates. For a detailed turn-level analysis examining the behavior of models trained with different RL algorithms, including metrics on average turn performance and history-best score accumulation, see Appendix F.2.

We attribute SAAC's training stability and effectiveness to TL-GRPO's fine-grained optimization strategy: by independently optimizing each turn within a trajectory, it provides the most detailed learning signal possible. In contrast, Trajectory-Level GRPO assigns a uniform advantage estimate to all tokens in one sequence, which introduces significant noise and leads to the observed instability and reward oscillations.

F.2 Analysis of Optimization Dynamics Across Turns

We analyze the evaluation behavior of models trained with the three RL algorithms to understand how they utilize the available optimization turns. Figure 5 presents two key metrics measured on out-of-domain tasks: (1) the average score achieved at each turn, and (2) the history-best score (i.e., the maximum reward observed up to that turn).

Three key observations emerge from this analysis:

Effectiveness of Initial Optimization: All three methods show a substantial increase in average turn score from turn 0 (initial random parameters) to turn 1, demonstrating that even a single optimization step guided by LLM reasoning significantly improves design quality. This immediate improvement distinguishes reasoning-guided iterative optimization from black-box methods, which typically require multiple exploratory samples before achieving comparable gains.

Exploration in Later Turns: After the first turn, average turn scores plateau or exhibit minor fluctuations, indicating that subsequent turns are primarily used for local exploration and refinement rather than consistently achieving higher scores. This pattern reflects a realistic balance between exploitation of current knowledge and exploration of the design space.

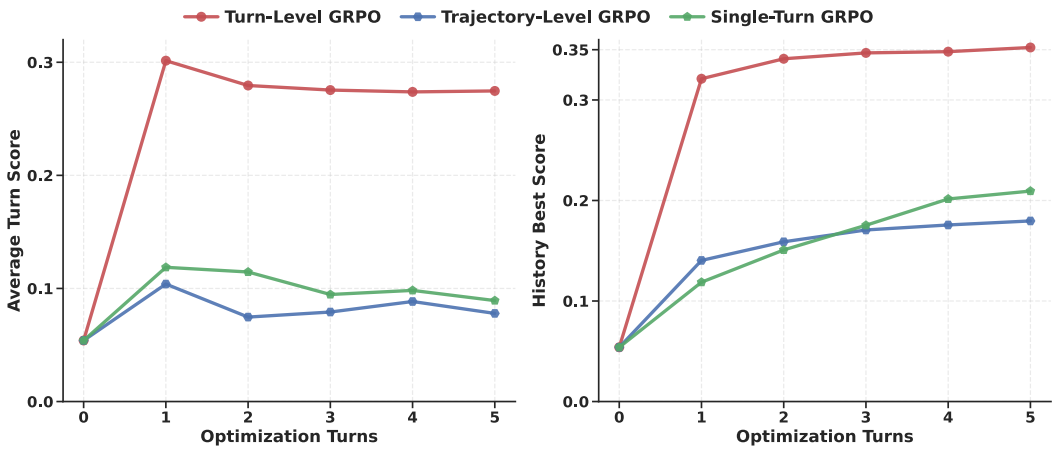


Figure 5: Comparison of turn-level performance for models trained with three RL algorithms on out-of-domain tasks. The turn-0 score represents the performance of initial values. Each line shows the average across all evaluation data.

Progressive Improvement in Best Design: Despite the plateau in average turn scores, the history-best score consistently improves across turns for all methods, confirming that additional optimization attempts increase the probability of discovering better designs. Notably, TL-GRPO achieves both higher average turn scores and faster accumulation of best scores throughout the optimization process.

These observations validate that TL-GRPO's turn-level optimization strategy effectively guides the agent to make higher-quality proposals at each step, leading to more efficient discovery of optimal designs within the fixed turn budget.

G Algorithm Details of TL-GRPO

We demonstrated the complete pseudocode for TL-GRPO in Algorithm 1, which serves as the core optimization engine within the SAAC framework. The algorithm formalizes the turn-level group sampling strategy and history-conditioned turn-level advantage estimation introduced in Sec. 3.1.

Algorithm 1 Turn-level GRPO

Input: Initial policy $\pi_{\theta_{\text{old}}}$, unified reward function R, dataset $\mathcal{D}$, group size G, max turns T, clip parameters ε
Output: π_θ
for each training iteration **do**
  Update old policy: $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$
  Sample a batch of input queries $\mathcal{Q}_b \sim \mathcal{D}$
  ▷ Asynchronous rollout phase
  for each query $q \in \mathcal{Q}_b$ **do**
    Generate one full trajectory
      $\tau = (o_0, a_0, o_1, \ldots, o_{T-1}, a_{T-1}, o_T) \sim \pi_{\theta_{\text{old}}}(\cdot \mid q)$
    Split τ at each turn to obtain history contexts:
      $\{h_t\}_{t=0}^{T-1}$ where $h_t = (q, o_0, a_0, \ldots, o_t)$
    for $t = 0$ **to** $T-1$ **do**
      Sampling $G-1$ actions $\{a_t^{(i)}\}_{i=1}^{G-1} \sim \pi_{\theta_{\text{old}}}(\cdot \mid h_t)$
      Compute turn-level rewards $\{r_t^{(i)}\}_{i=1}^{G}$
      Calculate turn-level advantage $A_t^{(i)}$ (Eq. 13)
    end for
  end for
  ▷ Collect all turn-level rollouts into one batch
  Update policy model π_θ by maximizing objective $\mathcal{J}_{\text{TL-GRPO}}(\theta)$ (Eq. 12)
end for

H DATA EXAMPLES

We provide an example of a synthetic OOD evaluation query (Figure 7), the tool schema used for simulation (Figure 6), and one selected intermediate response from our 30B model trained with TL-GRPO (Figure 8). In the selected turn, the model uses the preceding simulator feedback to target the gain and bandwidth bottlenecks, improving both while maintaining the power and phase-margin constraints; some reasoning is omitted for space.

Example Schema of Analog Circuit Simulation Tool

```
tool_schema:
  type: "function"
  function:
    name: "acs_simulate"
    description: |
      Simulates an analog circuit using Cadence Spectre.
      The function accepts a dictionary of transistor parameters. Depending on the specific
circuit topology,these may include widths (e.g., "w1", "w2", ...), lengths (e.g., "l1", "l2
", ...), custom-named sizing parameters (e.g., WBOOTPM1", ...), or passive component
parameters such as capacitors and resistors. The exact set and naming of available
parameters are determined by the circuit netlist. Each parameter value must be a string with
 appropriate unit. Please use "u" for sizing parameters, "k"(kilo-ohm) for resistors, and "p
"(pF) for capacitors. For example, use "2u" for 2 micrometers, "10k" for 10 kilo-ohms, and
"5p" for 5 picofarads.
      The simulation runs in an isolated temporary directory. Upon completion, it may
extract various performance metrics, which could include but are not limited to:
      **DC/Bias Metrics:**
      - vout: Output voltage in Volts
      **Frequency Response:**
      - gain: DC small-signal gain in dB
      - gbw: unity-gain bandwidth in Hz
      - pm: phase margin in degree
      - gm: Gain margin in dB
      - PhaseMarginFrequency: Frequency at phase margin in Mhz
      - GainMarginFrequency: Frequency at gain margin in Mhz
      **Power Supply Rejection (PSR):**
      - PSR@specific_freq: PSR at a specific frequency in dB(e.g., PSR@10Hz, PSR@10kHz
      - PSR_peak: Peak PSR in dB
      **Noise Performance:**
      - irn: input reference noise in Voltage
      - OutNoise@specific_freq: Output noise at a specific frequency in Voltage/sqrt(Hz)
      **Power Consumption:**
      - pw: total power consumption in Watts
      **Data Converter Metrics (if applicable):**
      - enob: effective number of bits (bits)
      - sfdr: spurious-free dynamic range (dBc)
      - snr: signal-to-noise ratio (dB)
      Returns a dictionary with the available simulation metrics as strings. The specific
metrics returned depend on the circuit type and simulation setup.

    parameters:
      properties:
        variables:
          type: "object"
          description: |
            A dictionary mapping transistor parameter names to their values as strings with
units. The available parameters depend on the circuit topology and may include: Generic
width/length names (e.g., "w1", "w2", ..., "l1", "l2", ...), Custom netlist-defined names (e
.g., "WCOMPN0", "WBOOTPM1", "WDACNM2", "FinWidth_Nbias0", etc.), Passive component
parameters (e.g., Cap_z3, Res_bias1) if applicable.
            Example with generic names: {"w1": "2u", "w2": "3u", ..., "wN": "4u"}
            The specific parameter names and their count are determined by the circuit
netlist. All values must be strings with appropriate units, not numbers.
```

Figure 6: Example tool schema for analog circuit simulation.

Example Synthetic Input Query with Initial Points (score = 0)

````
You are a senior analog integrated circuit design engineer responsible for performing a
transistor sizing optimization (Analog Circuit Sizing) task.
## Task Description
Your goal is to optimize the width or length parameters of all specified MOSFET transistors
in a given circuit netlist through iterative simulation and parameter adjustment, so that
they meet a series of preset performance metric requirements.
## Circuit Netlist
The following is the SPICE netlist description of the circuit, which defines the circuit
topology, component connections, and the parameters you need to optimize:
```spice
// This circuit should be designed and optimized based on the TSMC 40nm Mixed Signal RF
SALICIDE process library. In the circuit netlist, `nch` represents the NMOS transistor under
 this process with a DC bias voltage of 1.8V, and `pch` represents the PMOS transistor under
 this process with a DC bias voltage of 1.8V.
subckt opamp net_vb_n net_vb_p net_vin net_vip net_vo vdd vss
    I0_M5 (net_vb_n net_vb_n vss vss) nch l=120n w=w1 m=1 nf=4
    I0_M4 (net4 net_vb_n vss vss) nch l=120n w=w2 m=1 nf=4
    I0_M1 (net5 net_vin net4 vss) nch l=120n w=w3 m=1 nf=4
    I0_M0 (net_1 net_vip net4 vss) nch l=120n w=w4 m=1 nf=4
    I0_M3 (net5 net5 vdd vdd) pch l=120n w=w5 m=1 nf=4
    I0_M2 (net_1 net5 vdd vdd) pch l=120n w=w6 m=1 nf=4
    I1_M4 (net_vb_p net_vb_p vdd vdd) pch l=120n w=w7 m=1 nf=4
    I1_M1 (net_2 net70 vss vss) nch l=120n w=w8 m=1 nf=4
    I1_M0 (net70 net70 vss vss) nch l=120n w=w9 m=1 nf=4
    I1_M3 (net_2 net_vb_p vdd vdd) pch l=120n w=w10 m=1 nf=4
    I1_M2 (net70 net_1 vdd vdd) pch l=120n w=w11 m=1 nf=4
    I2_M2 (net_vb_n net_vb_n vss vss) nch l=120n w=w12 m=1 nf=4
    I2_M0 (net_vo net_vb_n vss vss) nch l=120n w=w13 m=1 nf=4
    I2_M1 (net_vo net_2 vdd vdd) pch l=120n w=w14 m=1 nf=4
    I3_M4 (net_vb_p net_vb_p vdd vdd) pch l=120n w=w15 m=1 nf=4
    I3_M1 (net_vo net71 vss vss) nch l=120n w=w16 m=1 nf=4
    I3_M0 (net71 net71 vss vss) nch l=120n w=w17 m=1 nf=4
    I3_M3 (net_vo net_vb_p vdd vdd) pch l=120n w=w18 m=1 nf=4
    I3_M2 (net71 net_1 vdd vdd) pch l=120n w=w19 m=1 nf=4
ends opamp
```
## Parameters to Optimize
The parameters to be optimized for this circuit are all width parameters represented by w1,
w2, ... w19. The value range for each parameter is from 0.9 µm to 9 µm.

## Initial Parameter Values
w1: 5.9 µm w2: 5.99 µm w3: 1.36 µm w4: 1.43 µm w5: 5.25 µm w6: 7.03 µm w7: 8.4 µm w8: 6.54 µ
m w9: 2.99 µm w10: 1.2 µm w11: 7.11 µm w12: 5.42 µm w13: 3.17 µm w14: 5.94 µm w15: 5.07 µm
w16: 2.05 µm w17: 1.5 µm w18: 2.77 µm w19: 7.48 µm

## Performance Targets
The optimized circuit must meet the following performance metrics:
- Gain > 79.14 dB
- Gain-Bandwidth Product (GBW) > 0.59 MHz
- Total Power Consumption (Power) < 17.77 µW
- Phase Margin (PM) > 70.95 degrees
// ... (details about task format and specific requirements are omitted for brevity)
// Initial value with simulation result
Arguments: {"variables": {"w1": "5.9u", "w2": "5.99u", "w3": "1.36u", "w4": "1.43u", "w5":
"5.25u", "w6": "7.03u", "w7": "8.4u", "w8": "6.54u", "w9": "2.99u", "w10": "1.2u", "w11":
"7.11u", "w12": "5.42u", "w13": "3.17u", "w14": "5.94u", "w15": "5.07u", "w16": "2.05u", "
w17": "1.5u", "w18": "2.77u", "w19": "7.48u"}}
Simulation Result:
{'gain': '64.5553', 'gbw': '23314.7', 'pw': '6.29918e-06', 'pm': '89.5388'}
````

Figure 7: An example of our synthetic query with initial points.

Example LLM Response Turn 2 (score: $0.24 \rightarrow 0.42$)

```
<Analyze>
According to the simulation results after this parameter adjustment, the circuit performance
 has improved significantly compared to the initial values but still fails to meet the
design targets, requiring further optimization. The detailed analysis is as follows:
1.  **Gain**: The current gain is 68.4952 dB, still below the target of 79.14 dB, with a gap
 of about 10.64 dB. It is primarily determined by the ratio of the transconductance (gm) to
the output impedance (ro) of the differential pairs (M1-M2, M0-M3). Currently, w3 and w4 (
input transistors) have been adjusted to 6.5u, providing a slight increase, but their widths
 are still insufficient. Moreover, their matching with w5 and w6 (PMOS) is suboptimal,
leading to inadequate overall transconductance. Additionally, the sizes of current mirrors (
e.g., w1~w2) and tail current transistors (w12~w13) are set too small to support high gain.

2.  **Gain-Bandwidth Product (GBW)**: The current GBW is 93.85 kHz, still far from the
target of 0.59 MHz (590 kHz), with a gap of 496 kHz. It is mainly influenced by the dominant
 pole frequency (determined by the gate-source capacitance and transconductance of the input
 transistors). Although significantly improved from the initial value (23.3 kHz), the
related transistors (e.g., w3~w4) are still relatively small, and the overall load
capacitance and current configuration are not ideal. Furthermore, the small size of the
output stage (e.g., M13~M14) also limits speed.

3.  **Power Consumption**: The current power consumption is 9.54 µW, which is close to but
still within the target range (<17.77 µW), leaving a significant margin. This benefits from
reducing the sizes of several "tail current" transistors (e.g., w1~w2, w7~w8, w12~w13) to
around 1u-2u. However, the device current is not sufficiently amplified in critical paths (e
.g., M3~M4, M5~M6). Therefore, based on this, the sizes of main current paths should be
reasonably increased to improve gain and speed while ensuring total power does not exceed
the limit.

4.  **Phase Margin**: The current phase margin is 89.56°, higher than the target of 70.95°,
and is very robust. This indicates good stability, meaning there is sufficient margin
between the dominant and second poles, and the current circuit shows no significant
oscillation tendency. However, with the current parameter settings, this high margin comes "
at the cost of bandwidth," as the bandwidth remains too low, representing a case of "high
margin but low efficiency."

5.  **Analysis of Key Transistor Sizes**:
... (some output omitted due to space) ...

6.  **Optimization Direction Summary**:
    *   **Primary Objective**: Increase gain (+~10.6 dB) and GBW (+~496 kHz).
    *   **Main Approach**: Increase the sizes of w3, w4 (differential input pair) and w5, w6
 (symmetric PMOS), while adjusting w1~w2 and w12~w13 (tail current) to coordinate the
improvement.
    *   **Secondary Objective**: Continue optimizing the size ratios of key transistors,
especially w13~w14, to enhance their drive capability for the output stage, without
significantly increasing power consumption (currently 9.54 µW).
    *   **Third Objective**: Adjust the reasonable allocation of various current mirror
ratios (e.g., w7~w10, w15~w17) to ensure they provide sufficient "pumping" current for
critical paths, while avoiding setting their sizes too small and dragging down overall
performance.

In summary, although the current performance is better than the initial state, substantial
size optimization is still needed, especially strategic reconfiguration of the differential
pair (w3~w6), current mirrors (w5~w6, w13~w14), and tail current structure (w1~w2, w12~w13).
 Through this round of adjustments, we can begin to approach the target while maintaining
the robustness of the phase margin.
</Analyze>
<tool_call>
Function: acs_simulate
Arguments: {"variables": {"w1": "1.0u", "w2": "1.0u", "w3": "8.0u", "w4": "8.0u", "w5": "7.4
u", "w6": "7.4u", "w7": "1.5u", "w8": "1.3u", "w9": "0.9u", "w10": "1.5u", "w11": "1.5u", "
w12": "1.0u", "w13": "3.7u", "w14": "3.7u", "w15": "1.5u", "w16": "1.5u", "w17": "0.9u", "
w18": "1.7u", "w19": "1.7u"}}
</tool_call>
<tool_response>
Simulation Result:
{'gain': '69.948', 'gbw': '166584', 'pw': '1.08715e-05', 'pm': '88.9833'}
```

Figure 8: An example of LLM response: Turn 2.

I LIMITATIONS

Our work has several limitations that point to directions for future research:

Resource constraints. While we compare SAAC with Bayesian optimization (BO) on analog circuit sizing, both methods operate under a limited simulation budget. This constraint is due to the computational expense of circuit simulation, where a single evaluation can take minutes to hours for complex circuits. Although we selected tasks with relatively fast simulations, training still requires significant computational resources. The comparison with BO under this limited budget remains valid, as we restrict the parameter search ranges to physically reasonable intervals, making the optimization tractable. Future work with increased simulation budgets would further validate our approach.

Turn-level reward design. While our unified, verifiable reward function effectively normalizes multiple objectives and provides a preference signal across actions, it may not be theoretically optimal. Furthermore, it may not fully capture the value of exploratory actions that may lead to better designs in later turns. Future work could explore more sophisticated reward calculation methods that better balance immediate performance with long-term exploration.

Dataset scale and diversity. We evaluated SAAC on 12 analog circuit tasks, which, while diverse, represent only a subset of real-world optimization problems. The high commercial value and confidentiality of industrial circuit designs limit data availability. Collecting more tasks would improve generalization. Additionally, we plan to extend our evaluation to other scientific optimization domains to further validate the broad applicability of our method.