2026 IEEE International Symposium on Circuits and Systems (ISCAS) | 979-8-3315-7769-8/26/$31.00 ©2026 IEEE | DOI: 10.1109/ISCAS66217.2026.11562451

Balancing Speed and Accuracy for Robust Analog-Mixed Signal Circuit Design using Closed-Loop Reinforcement Learning with Ensemble Neural Network Surrogates

Zuwei Guo*, Jie Fu*, Sumukh Bhanushali*, Zehua Zeng†, Imon Banerjee‡, Arindam Sanyal*
*Arizona State University, Tempe, AZ 85281, USA
†University of Maryland, College Park, MD 20742, USA
‡Mayo Clinic, Scottsdale, AZ 85259, USA
Email: arindam.sanyal@asu.edu

***Abstract*—Analog/mixed-signal (AMS) circuit sizing requires exploring high-dimensional, constrained design spaces under expensive SPICE simulations, while maintaining robustness to out-of-distribution (OOD) parameter proposals. We propose a fidelity-aware reinforcement learning framework (RL-PPO-Online) that unifies (i) SPICE-in-the-loop verification and (ii) ensemble neural network guided regularization for surrogate-assisted optimization. Another PPO policy (RL-PPO-Offline) is trained using an ensemble neural network for fast performance prediction, while a discriminator penalizes OOD actions to stabilize surrogate-only updates. We further introduce RL-PPO-Blend, which schedules a small fraction of SPICE queries (0.2–5%) on top of RL-PPO-Offline to provide a tunable speed–fidelity trade-off. On challenging ADC front-end sizing tasks in 28 nm (bootstrapped sampling switch and input buffer + switch), the proposed method improves pass rates over DDPG and LLM baselines, and achieves near-online accuracy with substantially reduced simulation cost.**

***Index Terms*—Reinforcement Learning, Proximal Policy Optimization (PPO), Discriminator Regularization, Analog-to-Digital Converter (ADC)**

I. Introduction

Analog/mixed-signal (AMS) sizing is labor-intensive due to nonlinear device physics, layout parasitics, and costly SPICE simulations. Learning-based approaches help, but many either overuse simulations (which are slow) or rely on surrogates without controlling for distribution shifts. We target high SNDR ADC front ends with an input buffer driving a bootstrapped sampling switch in 28 nm.

Conventional automation, knowledge-based [1]–[3], equation based [4], and simulation-based [5], [6] approachs face scalability or accuracy limits: rule/analytic sizing is time-consuming and poorly transferable [7], [8]; equation models oversimplify behavior; and simulation-driven optimization, while accurate, is computationally expensive.

Recent work applies ML/reinforcement learning (RL) to analog sizing and topology. AutoCkt uses discrete RL with layout-aware rewards [9]; MA-Opt and related work adopt multi-agent RL/DDPG for modular design [10], [11]; INSIGHT trains a transformer surrogate via PPO to reduce SPICE calls [12]; and ROSE-Opt mixes RL with Bayesian optimization [3]. GNNs exploit circuit graphs for topology/sizing [13], [14]. LLM-based tools generate code/topologies or guide Bayesian search [15]–[18] but lack tight physical feedback, which limits precision. A fidelity-aware RL that operates both in-loop and offline for complex AMS blocks—remains underexplored.

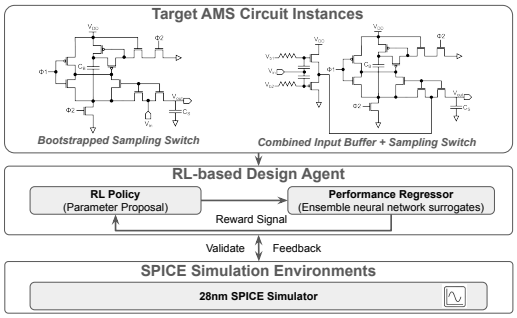

Fig. 1: Framework overview. A PPO policy uses an ensemble regressor for fast surrogate feedback; SPICE simulation is periodically called for accurate feedback, while an discriminator penalizes out-of-distribution actions. We evaluate two circuits: bootstrapped sampling switch and coupling it with a input buffer. The outcome represent high-linearity ADC front-ends with strict SNDR/SFDR.

In this work, we introduce a unified reinforcement learning framework for AMS circuit design that offers high-fidelity and fast optimization. Our key contributions include:

(1) a dual-mode reinforcement learning framework that supports both offline discriminator-guided and online SPICE-in-the-loop training, balancing optimization speed and fidelity.
(2) a discriminator-based critic that constrains action generation to in-distribution regions improves sample validity and training stability.
(3) a blend-mode that integrates online and offline training with a simple schedule that invokes SPICE on a small percentage of steps to trade runtime for accuracy without changing the PPO backbone.
(4) apply our method to two high-linearity ADC front-end circuits—an input buffer and a bootstrapped sampling switch—demonstrating reliable performance under stringent SNDR and SFDR targets.

II. Related Work

Automating analog design is challenging due to high-dimensional, nonlinear parameter spaces and costly SPICE loops. RL-based sizing/topology methods like AutoCkt use discrete policies with layout-aware rewards but remain simulation-heavy and scale poorly beyond small amplifiers [9]; multi-agent TD3/MA-Opt improves exploration and stability yet depends on predefined partitions and repeated simulations without in-loop feedback [10], [11]. Among RL algorithms, DDPG offers sample efficiency but is unstable and reward-sensitive [19], whereas PPO improves robustness via clipped objectives [20]. Surrogate/neural simulators reduce SPICE costs—INSIGHT trains a transformer via PPO for high-fidelity, low-latency prediction but still needs an external optimizer [12]. LLMs support sizing/code/topology (AnalogCoder, LaMAGIC) and datasheet-driven sizing via multi-agent LLMs [15], [16], [21], yet they often lack tight physical feedback; our LLaMA-2-7B baseline , for example, produced weak, physically ungrounded parameters. In contrast, our framework uses an ensemble regressor alongside PPO for feedback and supports both online SPICE-in-the-loop and offline training with a discriminator critic, enabling a tunable fidelity–efficiency trade-off and more reliable parameter generation.

III. Methodology

A. Problem Formulation

We formulate analog circuit design as a sequential decision-making problem, where the goal is to discover optimal circuit parameters $\mathbf{a} \in \mathbb{R}^d$ that meet target performance metrics: SNDR and SFDR. Given the non-linear, black-box nature of circuit behavior, we employ RL to learn a parameter generation policy $\pi_\theta(\mathbf{a}|\mathbf{s})$ that maximizes a performance-based reward to meet the target.

B. Framework Overview

Our framework consists of three main components: (1) an ensemble regressor trained to predict SNDR and SFDR values given circuit parameters, (2) an RL agent that learns to generate parameters, and (3) a discriminator-based critic that penalizes unrealistic or out-of-distribution parameters. Three operating modes are supported: an *offline mode* using only the regressor and discriminator-based guidance for fast training; an *online mode* that periodically invokes SPICE simulation for accurate performance evaluation and feedback; and a *blend mode* that mixes surrogate-guided updates with a scheduled fraction $p\%$ of SPICE-verified steps per epoch. We report $p \in \{0.2, 0.5, 1, 2, 5\}$.

C. Regressor for Performance Prediction

In offline, online and blend training modes, our framework employs a surrogate regressor to approximate key performance metrics: SNDR and SFDR. The regressor not only provides a fast reward estimation mechanism but also significantly reduces dependence on expensive SPICE simulations. The regressor is implemented as an ensemble of fully connected neural networks with two branches to predict SNDR and SFDR, given a set of circuit parameters $\mathbf{a}$. This ensemble modeling approach provides robustness against overfitting.

Unlike standard MSE, we use a regression focal loss to handle skewed errors by down-weighting easy cases and emphasizing hard, high-error samples:

$$\mathcal{L}_{\text{focal}} = \frac{1}{N}\sum_{i=1}^{N}\Big(1 - e^{-\|\mathcal{R}_i(\mathbf{a})-\mathbf{y}\|_2^2}\Big)^{\gamma}\|\mathcal{R}_i(\mathbf{a}) - \mathbf{y}\|_2^2, \quad (1)$$

where $\mathcal{R}_i(\mathbf{a})$ predicts $(\text{SNDR}, \text{SFDR})$, $\mathbf{y}$ is the SPICE simulated ground truth, and γ=2. In offline mode, the surrogate regressors are pre-trained and frozen during PPO; In online mode, they are incrementally updated with new SPICE verified data. This improves the fit in difficult cases and yields more reliable rewards for RL.

D. Reward Calculation using regressive model

To efficiently guide training, we construct the reward signal using predicted performance estimates from the ensemble regressors. For a generated parameter $\mathbf{a}$, the regressor predicts $(\hat{y}_{\text{SNDR}}, \hat{y}_{\text{SFDR}})$, and the reward is defined as the shortfall from target thresholds to stabilize learning:

$$R_{\text{pred}} = \min\left(\frac{\hat{y}_{\text{SNDR}} - \tau_{\text{SNDR}}}{\hat{y}_{\text{SNDR}} + \tau_{\text{SNDR}}}, 0\right) + \min\left(\frac{\hat{y}_{\text{SFDR}} - \tau_{\text{SFDR}}}{\hat{y}_{\text{SFDR}} + \tau_{\text{SFDR}}}, 0\right) \quad (2)$$

where τ_{SNDR} and τ_{SFDR} are the desired thresholds of performance that can be tuned on a case-by-case basis.

This non-positive reward formulation avoids saturation near threshold regions and provides smoother gradients for stable PPO updates.

E. SPICE-in-the-loop Optimization

In the online mode, we periodically replace regressor-based feedback with actual SPICE simulations. This provides high-fidelity supervision, mitigates reward drift, and enables long-term robustness. The simulation is invoked every K episodes (K can be changed based on time-efficiency requirements), and the RL agent receives real performance values for a batch of actions, which are used to both compute the reward and update the regressor.

F. Additional Critic to prevent distribution shift

To curb out-of-distribution (OOD) actions and overfitting, we add a discriminator D trained to separate real (SPICE) from generated parameters via BCE:

$$\mathcal{L}_{\text{disc}} = -\mathbb{E}_{x\sim\mathcal{D}_{\text{real}}}[\log D(x)] - \mathbb{E}_{x\sim\mathcal{D}_{\text{fake}}}[\log(1 - D(x))]. \quad (3)$$

We feed back its signal by penalizing confidence away from 0.5 (decision boundary):

$$R_{\text{dis}} = -\lambda\big(D_\phi(\mathbf{a}) - 0.5\big)^2, \qquad R_{\text{total}} = R_{\text{pred}} + R_{\text{dis}}. \quad (4)$$

This constrains exploration to the data manifold, mitigating OOD drift and PPO overfitting without additional SPICE calls.

G. Reinforcement Learning Formulation

We adopt PPO for stable and efficient continuous control.

Action: $\mathbf{a} \in [0, 1]^d$ encodes normalized circuit parameters (dimension d depends on the circuit).

State: $\mathbf{s}$ contains normalized targets (e.g., SNDR/SFDR) and optional constraints (e.g., Fs). Although the steps are single-shot,

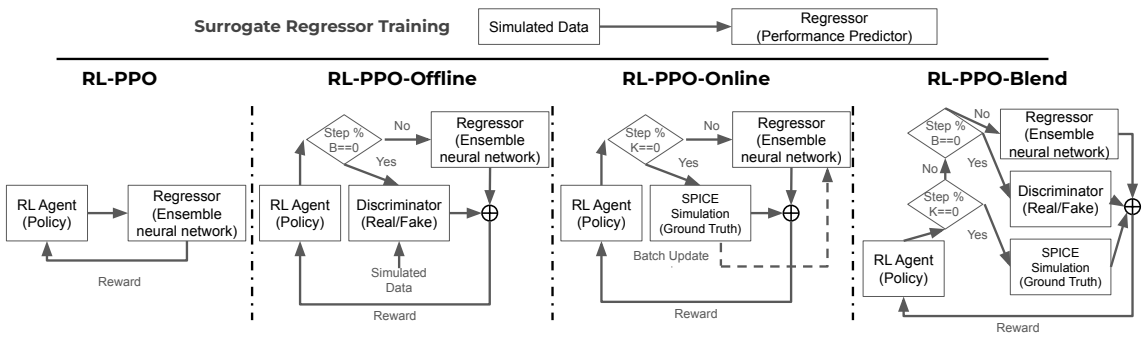

Fig. 2: Illustration of our RL framework. PPO-Offline uses surrogate rewards plus a discriminator penalty; PPO-Online alternates surrogate evaluation with periodic SPICE simulation for both RL feedback and retrain the regressor. The setup switches between online/offline to trade fidelity for efficiency. Here, K is the SPICE interval (every K steps, run SPICE for feedback and update the regressor) and B is the discriminator batch size.

s allow the policy to adapt to goals; richer embeddings (e.g., topology-aware) are left for future work.

Reward: Computed from the ensemble regressor or SPICE feedback; an optional discriminator penalty applies in offline/Blend modes.

Training Modes: Offline: fast learning via the frozen surrogate and discriminator. Online: periodic SPICE simulations attain rewards and retrain the surrogate regressors, aligning them with real behavior. Alternating these yields a tunable efficiency–fidelity trade-off and improves policy reliability over time.

IV. Component Importance

We quantify feature/component influence using the ensemble regressor's first-layer coefficients. Let $\mathbf{w}^{(m)}_{\mathrm{SNDR}} \in \mathbb{R}^d$ and $\mathbf{w}^{(m)}_{\mathrm{SFDR}} \in \mathbb{R}^d$ be the first-layer weights for the regressor m. For feature j:

$$I^{(m)}_{j,\mathrm{SNDR}} = \frac{|w^{(m)}_{\mathrm{SNDR},j}|}{\sum_k |w^{(m)}_{\mathrm{SNDR},k}|}, \qquad I^{(m)}_{j,\mathrm{SFDR}} = \frac{|w^{(m)}_{\mathrm{SFDR},j}|}{\sum_k |w^{(m)}_{\mathrm{SFDR},k}|}. \tag{5}$$

and the ensemble-averaged score

$$I_j = \tfrac{1}{2M} \sum_{m=1}^{M} \Big(I^{(m)}_{j,\mathrm{SNDR}} + I^{(m)}_{j,\mathrm{SFDR}} \Big). \tag{6}$$

We rank features by I_j (ties are broken by the SNDR channel). This approach is scale-invariant and uses weights only.

Findings (28 nm, buffer+switch). Top device-level drivers are `samp_nmos`, `nmos4`, `nmos5`, `inv_pmos`, `nmos2`, `pmos1`, `nmos3`, `nmos1`, `inv_nmos`, `pmos2`;

system-level: F_s (sampling frequency), C_B (bootstrap capacitor), `source_res`, `duty_cycle`, and C_S (sampling capacitance).

As expected, the sampling switch size is the most important feature for SNDR followed by C_B and input source resistance. A larger sampling switch reduces on-resistance and allows faster settling which improves distortion. The bootstrap capacitor forms a capacitive divider at the gate of the sampling switch with its parasitic capacitor, and C_B needs to scale with the sampling switch size to reduce distortion from parasitic capacitors. Increase in source resistance linearizes the overall resistance seen by the sampling switch and reduces distortion. However, at higher sampling frequencies, a large source resistance will reduce SNDR. The sampling capacitance C_S has a lower feature importance since it is set deterministically to meet the kT/C specification and does not limit SNDR in our experiments.

TABLE I: Component importance by ensemble weight score I_j (normalized).

Rank	Feature / Comp.	Rel. Score	Rank	Feature / Comp.	Rel. Score
1	samp_nmos	0.106	9	nmos3	0.063
2	CB	0.090	10	nmos1	0.060
3	source_res	0.077	11	inv_nmos	0.057
4	nmos4	0.070	12	pmos2	0.054
5	nmos5	0.068	13	Fs	0.046
6	inv_pmos	0.066	14	duty_cycle	0.042
7	nmos2	0.064	15	fbin	0.041
8	pmos1	0.064	16	CS	0.033

V. Experiments

A. Setup

We evaluate our RL framework on two demanding ADC front-ends: a bootstrapped sampling switch and coupling it with an input buffer (Fig. 1). We implement both designs in 28 nm CMOS, with 26 and 38 tunable parameters per circuit.

We compare multiple baselines. For instance, DDPG [19], and LLM modeling including few-shot (LLM-FS) and fine-tuned (LLM-FT) LLaMA-2-7B [22] against RL-PPO. We also setup comparisons of offline and online training, for their speed and fidelity (pass rate), and a blend mode for which we sweep the SPICE fraction $p\%$ in $\{0.2, 0.5, 1, 2, 5\}$, to compare the trade-off between speed and fidelity.

B. Training Details

We use PPO as the base RL algorithm. The policy and value networks share a 2-layer MLP architecture with hidden sizes [64,64]. Training uses a learning rate of 1×10^{-4}, $\gamma = 0.99$, entropy 0.05, batch size 64, and is trained for 10^5 steps, with early stop at zero cumulative reward. Rewards rely on a learned surrogate to keep actions valid.

Offline circuit performance is approximated by an ensemble of regressors predicting SNDR/SFDR from the generated parameters. Each base regressor is a 4-layer MLP with hidden sizes of [64, 32, 16, 8], with different activations and initializations, trained with Adam and dropout; ensemble means are averaged for robustness. The discriminator is trained every K steps with a batch of data of size 2K. Trained on 4,790 SPICE-labeled samples, the surrogate attains $R^2 = 0.88$ versus ground truth.

For the online setting, SPICE simulations are invoked every episode for parameter evaluation and data augmentation. The discriminator is trained concurrently in offline mode to distinguish real from generated samples. All regressors are incrementally retrained using new SPICE data collected during exploration in online mode. Hyperparameters were optimized through grid search. PPO is implemented via Stable-Baselines3, while the regressors and discriminator are built using TensorFlow. Circuit evaluations are handled by Cadence Spectre, automated through MATLAB and Python integration.

C. Performance Evaluation

We evaluate all baseline and proposed methods based on the percentage of generated parameter sets that achieve SNDR ≥ 60dB and SNDR ≥ 75dB, corresponding to 10-bit and 12-bit ADC designs, respectively, on unseen test points. Table II reports the performance across circuit types and data budget settings. For the input buffer + bootstrapped switch, RL-PPO attains $68.1\%/58.2\%$ (SNDR$\geq 60/75$ dB), outperforming RL-DDPG ($65.8\%/56.2\%$) and substantially exceeding LLM baselines (LLM-FS $28.6\%/11.6\%$, LLM-FT $27.8\%/10.5\%$). For the standalone bootstrapped switch, RL-PPO reaches $84.0\%/80.0\%$, again surpassing RL-DDPG ($81.4\%/73.5\%$) and both LLM variants (LLM-FS $40.7\%/16.9\%$, LLM-FT $38.2\%/24.8\%$). Overall, PPO provides higher pass rates and lower runtime than DDPG across circuits. Partial fine-tuning of the LLM using LoRA [23] underperformed compared to few-shot in-context learning, likely due to the limited training data, which may have hindered the model's ability to effectively generalize during fine-tuning.

TABLE II: Performance comparison across methods

Method	Circuits	SNDR ≥ 60 dB	SNDR ≥ 75 dB	Time (s) per 1k steps
LLM-FS	input buffer + bootstrapped sampling switch	28.6%	11.6%	N.A.
LLM-FT		27.8%	10.5%	N.A.
RL-DDPG		65.8%	56.2%	243
RL-PPO		68.1%	58.2%	167
LLM-FS	bootstrapped sampling switch	40.7%	16.9%	N.A.
LLM-FT		38.2%	24.8%	N.A.
RL-DDPG		81.4%	73.5%	216
RL-PPO		84.0%	80.0%	172

D. Balancing Speed and Fidelity

We compare two PPO variants, RL-PPO-Offline and RL-PPO-Online, in Table III (PPO variants), then study blended schedules that trade speed for fidelity. RL-PPO-Offline omits SPICE simulation during training, using an ensemble regressor plus a co-trained discriminator to guide the policy. It is $\sim 19\times$ faster than RL-PPO-Online while improving pass rates by $+9\%$ for SNDR ≥ 60 dB and $+12\%$ for SNDR ≥ 75 dB over the RL-PPO baseline. RL-PPO-Online inserts SPICE in the loop, achieving the best fidelity, $+15\%$ and $+18\%$ at the two thresholds over baseline, but at the cost of runtime ($\sim 19\times$ slower than RL-PPO-Offline).

TABLE III: Pass rates (SNDR thresholds) and time/1k steps for PPO variants. Blend-p uses SPICE for $p\%$ of steps while co-training the discriminator.

Method	Circuits	SNDR ≥ 60 dB	SNDR ≥ 75 dB	Time (s) per 1k steps
RL-PPO	input buffer + bootstrapped sampling switch	68.1%	58.2%	167
RL-PPO-Offline		77.6%	70.2%	174
RL-PPO-Online		82.8%	75.8%	3232 (×19)
RL-PPO-Blend-0.2		81.3%	74.3%	526
RL-PPO-Blend-0.5		84.7%	77.8%	987
RL-PPO-Blend-1		90.9%	88.0%	1686
RL-PPO-Blend-2		96.5%	92.3%	3480
RL-PPO-Blend-5		97.1%	94.3%	4999

To balance speed and fidelity, we introduce RL-PPO-Blend: while training with the discriminator to distinguish real from fake samples, we gradually increase the fraction of SPICE-verified steps per epoch, while the remaining steps utilize surrogate regressor feedback. A schedule value of 5 means that 5% of all training steps invokes SPICE simulation.

Across RL-PPO-Blend schedules (0.2 to 5), pass rates rise monotonically with the SPICE simulation fraction, while runtime increases accordingly. At 0.2%, RL-PPO-Blend reaches $81.3\%/74.3\%$ (SNDR$\geq$60/75 dB) using $\sim 16\%$ of RL-PPO-Online time (526 s vs. 3232 s). Notably, 0.5% already exceeds RL-PPO-Online on both thresholds (84.7% vs. 82.8%; 77.8% vs. 75.8%) at only $\sim 30\%$ of its runtime (987 s), and 1% offers a good balance at $90.9\%/88.0\%$ with $\sim 52\%$ of RL-PPO-Online time (1686 s). Higher fractions (2%–5%) push fidelity further (up to $97.1\%/94.3\%$) but with a steep increase in runtime (3480–4999 s), surpassing RL-PPO-Online's accuracy yet eroding its speed advantage. Overall, RL-PPO-Blend provides a tunable speed–fidelity trade-off; in our setup, 0.5–1% SPICE achieves better accuracy than RL-PPO-Online while remaining 2×–3× faster.

VI. Conclusion

This work presents a reinforcement learning framework for AMS circuit design that integrates online SPICE-in-the-loop simulation and offline discriminator-guided evaluation with a PPO-based policy, an ensemble of regressors, and a discriminator critic to efficiently generate high-quality parameter sets. We further introduce RL-PPO-Blend, which schedules a small fraction (0.2–5%) of SPICE-verified steps while co-training the discriminator, providing a tunable speed–fidelity trade-off and higher pass rates at modest runtime. Experiments show that our approach achieves higher pass rates than RL and LLM baselines on challenging ADC front-end blocks. The proposed framework enables scalable and automated analog design with reduced reliance on manual tuning and costly simulations, marking a step forward in practical AMS design automation. Future work will explore extending this framework to larger design spaces and incorporating layout-aware constraints.

REFERENCES

[1] W. Cao, M. Benosman, X. Zhang, and R. Ma, "Domain knowledge-based automated analog circuit design with deep reinforcement learning," *arXiv preprint arXiv:2202.13185*, 2022.

[2] A. Girardi, T. De-Oliveira, S. Ghissoni, P. C. Aguirre, and L. Compassi-Severo, "A comprehensive review on automation-based sizing techniques for analog ic design," *Journal of Integrated Circuits and Systems*, vol. 17, no. 3, pp. 1–14, 2022.

[3] W. Cao, J. Gao, T. Ma, R. Ma, M. Benosman, and X. Zhang, "Rose-opt: Robust and efficient analog circuit parameter optimization with knowledge-infused reinforcement learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[4] G. Zhang, H. He, and D. Katabi, "Circuit-gnn: Graph neural networks for distributed circuit design," in *International conference on machine learning*. PMLR, 2019, pp. 7364–7373.

[5] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolic, "Autockt: Deep reinforcement learning of analog circuit designs," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 490–495.

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

[7] M. Barari, H. R. Karimi, and F. Razaghian, "Analog circuit design optimization based on evolutionary algorithms," *Mathematical Problems in Engineering*, vol. 2014, no. 1, p. 593684, 2014.

[8] R. Rashid, G. Raghunath, V. Badugu, and N. Nambath, "Performance evaluation of evolutionary algorithms for analog integrated circuit design optimisation," *Microelectronics Journal*, vol. 141, p. 105983, 2023.

[9] K. Settaluri, Z. Liu, R. Khurana, A. Mirhaj, R. Jain, and B. Nikolic, "Automated design of analog circuits using reinforcement learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 9, pp. 2794–2807, 2021.

[10] Y. Choi, S. Park, M. Choi, K. Lee, and S. Kang, "Ma-opt: reinforcement learning-based analog circuit optimization using multi-actors," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2024.

[11] J. Zhang, J. Bao, Z. Huang, X. Zeng, and Y. Lu, "Automated design of complex analog circuits with multiagent based reinforcement learning," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[12] S. Poddar, Y. Oh, Y. Lai, H. Zhu, B. Hwang, and D. Z. Pan, "Insight: Universal neural simulator for analog circuits harnessing autoregressive transformers," *arXiv preprint arXiv:2407.07346*, 2024.

[13] A. Shahane, S. Swapna Manjiri, A. Jain, and S. Kumar, "Graph of circuits with gnn for exploring the optimal design space," *Advances in neural information processing systems*, vol. 36, pp. 6014–6025, 2023.

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

[15] Y. Lai, S. Lee, G. Chen, S. Poddar, M. Hu, D. Z. Pan, and P. Luo, "Analogcoder: Analog circuit design via training-free code generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 1, 2025, pp. 379–387.

[16] 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 preprint arXiv:2407.18269*, 2024.

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

[18] P. Li *et al.*, "Llm-uso: Large language model-based universal sizing optimizer," *arXiv preprint arXiv:2502.02764*, 2025.

[19] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[21] 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.

[22] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[23] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.