MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

ROSE-Opt: Robust and Efficient Analog Circuit Parameter Optimization with Knowledge-infused Reinforcement Learning

Cao, Weidong; Gao, Jian; Ma, Tianrui; Ma, Rui; Benosman, Mouhacine; Zhang, Xuan

TR2024-132 October 02, 2024

Abstract

Design automation of analog circuits has long been sought. However, achieving robust and efficient analog design automation continues to be a significant challenge. This paper proposes a learning framework, ROSE-Opt, to achieve robust and efficient analog circuit parameter optimization. ROSE-Opt has two important features. First, it incorporates key domain knowledge of analog circuit design, such as circuit topology, couplings between circuit specifications, and variations of process, voltage, and temperature, into the learning loop. This strategy facilitates the training of an artificial agent capable of achieving design goals by identifying device parameters that are optimal and robust. Second, it exploits a two-level optimization method, that is, integrating Bayesian optimization (BO) with reinforcement learning (RL) to improve training efficiency. In particular, BO is used for coarse search by quickly finding an initial starting point for optimization. This sets a solid foundation to efficiently train the RL agent with fewer samples. Experimental evaluations on circuit benchmarks show a promising sampling efficiency and an extraordinary figure of merit in terms of design efficiency and design success rate of our framework, as compared to prior methods. Furthermore, this work thoroughly studies the performance of different RL optimization algorithms, such as Deep Deterministic Policy Gradients (DDPG) with an off- policy learning mechanism and Proximal Policy Optimization (PPO) with an on-policy learning mechanism. This investigation provides users with guidance on choosing the appropriate RL algorithms to optimize the parameters of analog circuit devices. Finally, ROSE-Opt has also been studied to demonstrate promise in parasitic-aware device optimization for analog circuits. In summary, our work reports a knowledge-infused RL design automation framework for reliable and efficient optimization of analog circuits' device parameters. The codes of our method are open-sourced at https://github.com/xz-group/RoSE.

IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 2024

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

ROSE-Opt: Robust and Efficient Analog Circuit Parameter Optimization with Knowledge-infused Reinforcement Learning

Weidong Cao, *Member, IEEE,* Jian Gao, *Student Member, IEEE,* Tianrui Ma, *Student Member, IEEE,* Rui Ma, *Senior Member, IEEE,* Mouhacine Benosman, *Senior Member, IEEE,* and Xuan Zhang, *Senior Member, IEEE*

***Abstract*—Design automation of analog circuits has long been sought. However, achieving robust and efficient analog design automation continues to be a significant challenge. This paper proposes a learning framework, ROSE-Opt, to achieve robust and efficient analog circuit parameter optimization. ROSE-Opt has two important features. First, it incorporates key domain knowledge of analog circuit design, such as circuit topology, couplings between circuit specifications, and variations of process, voltage, and temperature, into the learning loop. This strategy facilitates the training of an artificial agent capable of achieving design goals by identifying device parameters that are optimal and robust. Second, it exploits a two-level optimization method, that is, integrating Bayesian optimization (BO) with reinforcement learning (RL) to improve training efficiency. In particular, BO is used for coarse search by quickly finding an initial starting point for optimization. This sets a solid foundation to efficiently train the RL agent with fewer samples. Experimental evaluations on circuit benchmarks show a promising sampling efficiency and an extraordinary figure of merit in terms of design efficiency and design success rate of our framework, as compared to prior methods. Furthermore, this work thoroughly studies the performance of different RL optimization algorithms, such as Deep Deterministic Policy Gradients (DDPG) with an off-policy learning mechanism and Proximal Policy Optimization (PPO) with an on-policy learning mechanism. This investigation provides users with guidance on choosing the appropriate RL algorithms to optimize the parameters of analog circuit devices. Finally, ROSE-Opt has also been studied to demonstrate promise in parasitic-aware device optimization for analog circuits. In summary, our work reports a knowledge-infused RL design automation framework for reliable and efficient optimization of analog circuits' device parameters. The codes of our method are open-sourced at https://github.com/xz-group/RoSE.**

W. Cao is with the Department of Electrical & Computer Engineering; The George Washington University, Washington, DC, 20052 USA, email: {weidong.cao@gwu.edu}. W. Cao was partly supported by the National Science Foundation under Grant no. CCF-1942900, and partly by Mitsubishi Electric Research Laboratories where he did his internship on this project.

J. Gao and X. Zhang are with the College of Engineering; Northeastern University, Boston, MA, 02115 USA, e-mail: {xuan.zhang@northeastern.edu}. J. Gao and X. Zhang were supported by the National Science Foundation under Grant no. CCF-1942900.

T. Ma is with the Department of Electrical and Systems Engineering; Washington University in St. Louis, St. Louis, MO, 63130 USA. T. Ma was supported by the National Science Foundation under Grant no. CCF-1942900.

M. Benosman is with the Mitsubishi Electric Research Laboratories, Cambridge, MA, 02139 USA, e-mail: {benosman@merl.com}. M. Benosman was solely supported by MERL.

R. Ma is with pSemi a Murata company, San Diego, CA, 92121 USA, e-mail: {ruimar@gmail.com}. R. Ma was solely supported by MERL.

I. Introduction

Integrated circuit (IC) technology advances human society by powering numerous applications and infrastructures with microelectronic chips of a small footprint. Recent advances in deep learning have shown great promise in transforming modern IC design workflows [1]–[3]. By formulating each design stage as a learning problem, machine learning techniques can significantly shorten IC development cycles compared to conventional Electronic Design Automation (EDA) tools. For example, Google [2] and Nvidia [3] have shown that deep learning methods can improve design efficiency by an order of $100\times$ at certain stages of the digital IC design flow, such as floor planning and power estimation. Analog circuit is an essential type of circuit that bridges our physical world with the digital information realm [4]–[8]. Yet, unlike digital ICs that benefit from well-established conventional EDA tools or emerging efficient learning-based design automation methods, analog circuits continue to rely on onerous human efforts and lack effective EDA techniques at all stages [1], [9].

Pre-layout design of analog circuits can be represented as a parameter-to-specification (P2S) optimization problem. Given the circuit topology, the goal is to find optimal device parameters (e.g., width and finger number of transistors) to meet the desired specifications (e.g., power and bandwidth) of the circuit. The problem is challenging due to several factors. First, it involves searching for parameters of diverse devices in a large design space. The complexity grows exponentially with an increase in both design parameters and circuit specifications [4], [5]. Second, the actual interactions between the device parameters and the circuit specifications are very complicated [1], [9], depending on multiple variables, such as the circuit topology, the variations in process, voltage, and temperature (PVT) and the parasitic effects of post-layout. There are no exact analytical rules to follow, which worsens the search process. Conventionally, human designers use critical domain knowledge, such as circuit topologies and couplings between circuit specifications, to manually derive the device parameters. In particular, a human designer exerts an intense effort to obtain empirical equations between the device parameters and the circuit specifications based on a simplified circuit topology. However, despite the simplification, tens and even hundreds of iterative fine-tunings are still required to ensure the design's accuracy and reliability.

During the past several decades, there have been enormous explorations on automating the design of analog circuit device parameters. These methods generally fall into two categories, knowledge-based techniques and optimization-based techniques. Knowledge-based techniques are designer-centric [10]–[12]. They customize the design steps for specific circuits based on domain knowledge and embed them into procedural scripts that mimic the actions of designers. These scripts allow designers to have full control over the modification and debugging of circuits to guarantee design reliability. However, design efficiency is significantly thwarted, because designers, acting as optimization agents, are required to frequently interact with procedural scripts. In contrast, optimization-based techniques are algorithm-centric. They consider each step of analog circuit design as a black-box optimization problem and use optimization methods, such as Bayesian optimization [1], Genetic algorithms [13], and emerging machine learning algorithms [9], [14]–[18] to address it. These algorithms can be run quickly to complete the design of an analog circuit with high efficiency. Unfortunately, due to the absence of knowledge from experienced designers, the reliability of the design is not guaranteed, e.g., device parameters are not robust to various non-idealities. These defects limit the efficiency and reliability of state-of-the-art analog design automation techniques.

To bridge this gap, we propose a learning framework, ROSE-Opt, to achieve robust and efficient analog circuit parameter optmization by synergizing domain knowledge of analog circuits and learning algorithms. Analog circuit design strongly relies on domain knowledge, such as circuit topology, couplings between circuit specifications, and PVT variations; thus, without adequately considering these key domain knowledge in building learning-based design automation frameworks, the device parameters discovered by the algorithm are prone to suffer from inferior reliability issues due to various non-idealities. Our previous work [19] follows this principle and has explored the integration of this key domain knowledge into the design framework. It also exploits a two-level optimization method by integrating Bayesian optimization (BO) and reinforcement learning (RL) to improve training efficiency.

In this paper, we propose the ROSE-Opt framework which advances the state-of-the-art method [19]. In particular: ❶ We analyze the failed cases in which our trained RL agent cannot converge to the optimal device parameters. In these scenarios, RL agent can still help designers by offering optimized initial points for manual tuning. ❷ We study the ability to consider device parasitics in parameter optimization. A direct mapping of an analog circuit schematic with correctly-sized devices into a physical layout can lead to performance degradation, mainly due to parasitics from metal wires and electromagnetic effects. Human experts often require tens of iterations between a schematic design and a physical layout design to find the final device parameters to ensure that the performance of post-layout simulation still satisfies the desired goals. This extended work demonstrates the promise of ROSE-Opt in addressing this problem. ❸ At the algorithm level, we thoroughly study the performance of different RL optimization algorithms, such as Deep Deterministic Policy Gradients (DDPG) with an off-policy learning mechanism and Proximal Policy Optimization (PPO) with an on-policy learning mechanism, to provide users with useful guidance in choosing appropriate RL algorithms for device sizing.

In summary, we present a comprehensive RL-based design automation framework to perform the P2S task of analog circuit design with high robustness and efficiency. We make these key contributions.

- This paper proposes a comprehensive RL-based design automation framework, ROSE-Opt, for robust and efficient optimization of device parameters in analog circuits. To achieve this goal, our learning framework sufficiently explores and exploits both domain knowledge of analog circuit design and the strong optimization ability of design automation algorithms.
- We perform failure analysis on our method and show how to leverage the unsuccessful deployment trajectory to guide the fine-tuning of manual efforts toward design success. In addition, we study its effectiveness in the scenarios of parasitic-aware device parameter optimization.
- At the algorithm level, we thoroughly study the performance of different RL optimization algorithms (i.e., DDPG vs. PPO) to provide users with useful insights in choosing appropriate RL algorithms for device parameter optimization.
- Experimental evaluations on circuit benchmarks show that our framework achieves $7.9\times \sim 12\times$ improvement in sampling efficiency and a significant improvement in design success rate, robustness, and reliability compared to the state-of-the-art methods.

The remainder of this paper is organized as follows. Section II provides the background context and related work. The proposed comprehensive RL framework is elaborated in Section III. The experimental methodology is described in Section IV. We present the results in Section V before concluding the paper in Section VI.

II. Background and Related Work

In this section, we first review the basics of Bayesian optimization and reinforcement learning. We then introduce the key domain knowledge that human experts commonly consider when addressing the P2S problem. Finally, we discuss existing design automation methodologies for analog circuits.

A. Bayesian Optimization

Bayesian optimization (BO) proves to be a valuable framework to address challenging black-box optimization problems that involve costly function evaluations. Fig. 1(a) shows an example of BO with two iterations ($t = 2$ and $t = 3$). BO's fundamental concept is to construct an inexpensive surrogate model, such as a Gaussian Process, by leveraging actual experimental data. This surrogate model incorporates prior knowledge or beliefs about the objective function, which is then used to make informed decisions in the process of selecting a sequence of function evaluations through the use of an acquisition function, such as expected improvement (EI). It also balances exploration and exploitation. Exploration allows

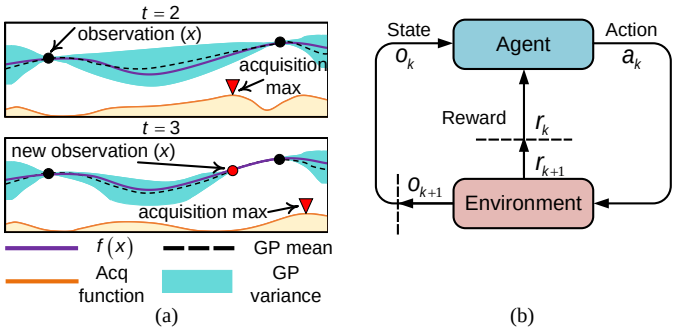

Fig. 1: (a) An illustration of Bayesian optimization to find the optima. Here, we use the Gaussian Process (GP) as the surrogate model and show two iterations. The plots show the mean and confidence intervals estimated with the GP model of the objective function, $f(x)$, which in practice is unknown. The plots also show the acquisition (Acq) functions in the lower-shaded plots. The acquisition is high where the model predicts a high objective (exploitation) and where the prediction uncertainty is high (exploration). (b) A simplified illustration of reinforcement learning. It includes five parts: agent, action, state, reward, and environment.

for a broader exploration of the search space, potentially discovering better solutions, while exploitation focuses on exploiting the known promising areas to optimize the current best solution. Balancing these two aspects is crucial to finding better solutions and refining the best solution.

Given an arbitrary function $f(\vec{x})$ for maximization, there are several steps to follow for BO. Step 1: initial sampling. Here a limited set of sample points is randomly selected. Step 2: initializing the model. These points from Step 1 are used to calculate a surrogate function. Step 3: iterating. In particular, the acquisition function is first used to get the next point; then, the surrogate function is re-evaluated; third, the surrogate function is verified to see if it remains stable or if the variance falls below a predetermined threshold, or if $f(\cdot)$ is exhausted, depending on the specific design objective.

BO is well suited to optimizing hyperparameters of many classification and regression models. It is also used to automate the P2S task of analog circuit design [1].

B. Reinforcement Learning

Reinforcement learning (RL) is a machine learning method related to how intelligent agents take actions in an environment to maximize cumulative returns based on states. As illustrated in Fig. 1(b), there are five essential elements in an RL problem: Agent, Action, State, Reward, and Environment. The 'Agent' is the learner and the decision maker who learns experiences from the training process and makes decisions based on observations (states) from the environment. The 'Action' is a set of operations that the agent can perform in a state. The 'State' is a representation of the current environment (i.e., observations) in which the agent is staying. This state can be observed by the agent and it includes all relevant information about the environment that the agent needs to know to make a decision. The 'Reward' is a scalar value returned by the environment after the agent takes an action in a state. It is used to evaluate and guide the actual learning behavior of the agent. The 'Environment' is the physical world in which the agent operates.

In each episode, an agent starts from an initial state, then observes the state o_k and takes an action a_k based on a policy. Meanwhile, the environment updates a reward r_{k+1} for that particular action and enters a new state o_{k+1}. The agent iterates through the episode in multiple steps, accumulating the reward at each step to obtain the final return. With multiple episodes, the RL agent improves its decision quality and finds the best policy to maximize the return. Such a policy would be deployed for practical tasks, i.e., the agent follows the trained policy to finish a given task.

RL algorithms have been extensively applied to many problems such as game playing [20], robotics [21], computer vision [22], and natural language processing [23]. RL has also been used to automate the design of ICs, such as the placement of the digital IC chip [24] and the P2S optimization of analog circuits [17], [18], [25].

C. Key Domain Knowledge of Analog Circuit Design

At the pre-layout stage, there are many considerations to be taken by human experts to ensure reliable device parameters to meet the design goals. These considerations are the domain knowledge, and we introduce the major ones that are commonly used by human experts when they tackle the P2S task, as shown in Fig. 2.

1) Circuit topology: When human experts manually find the optimal device parameters, they first construct the circuit small-signal model from the circuit topology, based on which they obtain empirical equations that connect device parameters to circuit specifications. With these equations, device parameters can be derived by hand.

2) Couplings between circuit specifications: Due to design trade-offs, circuit specifications often depend on each other. For example, in the design of operational amplifiers, energy efficiency often trades off with gain; that is, a higher amplification gain requires a larger transconductance, which, however, demands more power consumption and results in lower energy efficiency. Therefore, in a conventional manual design process, human experts use tens and even hundreds of iterative fine-tunings to find a group of proper device parameters to satisfy all circuit specifications.

3) PVT variations: To ensure the robustness of analog circuits in different harsh environments, a key design consideration is to minimize the influence of variations in process (P), voltage (V), and temperature (T). Process variation represents the deviation of the manufactured devices from their ideal specification due to manufacturing errors. It includes typical N-type transistor/typical P-type transistor (TT), fast N-type transistor/fast P-type transistor (FF), slow N-type transistor/slow P-type transistor (SS), slow N-type transistor/fast P-type transistor (SF), and fast N-type transistor/slow P-type transistor (FS). Voltage and temperature variations are due to uncertain ambient changes. Typical deviation on the supply voltage is $\pm 10\%$ from its nominal value V_{DD}; and typical range of the environmental temperature for circuits is $[-40, 125]^{\circ}C$. A single PVT corner is a combination of P, V, and T from their varying ranges. All these variations are unavoidable and can cause the circuit performance degeneration compared to its nominal case, i.e., $\{\mathrm{TT}, V_{\mathrm{DD}}, 25^{\circ}C\}$. Manual experts have to look for robust device parameters to achieve the design goal in all PVT corners.

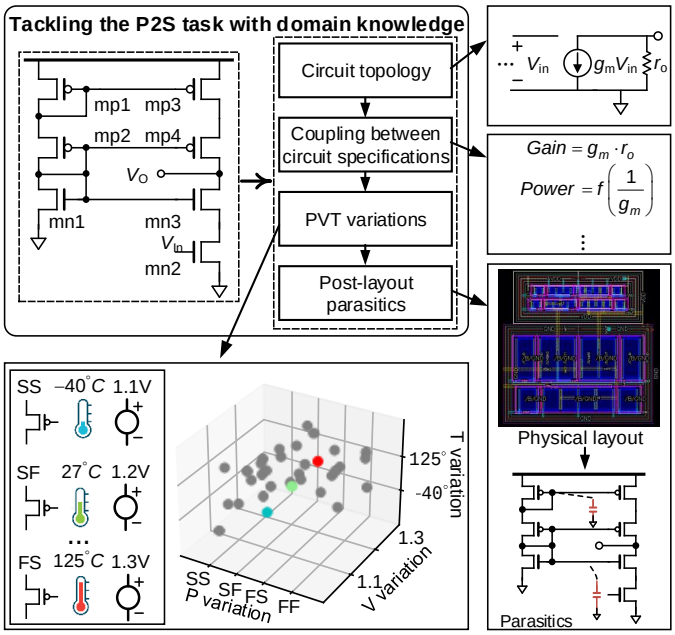

Fig. 2: Illustration of a manual design flow to tackle the P2S task with human domain knowledge.

4) Parasitic effects of physical layouts: A complete flow of analog circuit design includes the schematic design and the physical layout design. The conversion of an analog circuit schematic, with the correctly-sized components, into a physical layout can cause performance degradation due to the parasitic effects of metal wires and electromagnetic couplings. Experienced human designers often make efforts to adjust the device parameters to ensure that the post-layout simulation meets the desired objectives.

D. Existing Design Automation Methodologies

Various design automation techniques have been proposed for the P2S task of analog circuits in recent years. They generally fall into two categories: knowledge-based techniques and optimization-based techniques. Knowledge-based techniques, such as BAG [10], are designer-centric. They tailor the design steps for specific circuits with domain knowledge and embed these steps into the procedural scripts that mimic designer actions. These scripts provide designers with complete control over circuit modifications and debugging to ensure design reliability. Yet, these approaches notably affect design efficiency, as they demand frequent interactions between designers and procedural scripts, with designers playing the role of optimization agents. On the contrary, optimization-based methods such as BO [1], Geometric Programming [26], Genetic algorithms [13], and modern machine learning approaches [9], [14]–[18], [25] are centered on algorithms. They treat each step in a circuit design as a black-box optimization problem and can swiftly perform optimization procedures to complete a circuit's design with high efficiency. Unfortunately, the lack of knowledge from seasoned designers means that design reliability, such as the robustness of device parameters to non-ideal conditions, is not assured. These limitations significantly impact the widespread applications of state-of-the-art analog design automation techniques, as they are unable to achieve both high design efficiency and reliability.

Essential to advance analog design automation is to adequately incorporate analog design knowledge into optimization algorithms to ensure design reliability while not losing optimization efficiency. Learning-based optimization methods have recently emerged to show higher design efficiency in handling the P2S task compared to classical optimization algorithms such as BO [1], Geometric Programming [26], and Genetic algorithms [13]. As an example, supervised learning methods [9], [14]–[16] have been used to learn the complicated relations between device parameters and circuit specifications. Once trained, they adopt one-step inference to predict optimal device parameters for given design goals. Nonetheless, these supervised learning methods cannot guarantee a high design success rate and suffer from weak generalization abilities [9], [14]–[16] due to their inherent approximation errors.

On the other hand, RL methods [17], [18], [25] learn an optimal policy from the state space of circuit specifications to the action space of device parameters, which solves a quasi-dynamic programming problem. They often use multiple sequential decision steps to find the optimal device parameters rather than just using one-step prediction, thus achieving a higher design success rate and better generalization abilities than supervised learning methods [9], [14]–[16]. However, none of these learning algorithms has taken into account sufficient domain knowledge of analog circuit design in the optimization loop, leading to low design reliability.

In this work, we propose a learning-based framework, ROSE-Opt, to achieve efficient and reliable parameter optimization of analog circuit devices by harnessing the synergy between the knowledge of human designers and RL algorithms (elaborated in Section III). In particular, we leverage the rapid convergence of BO to identify an optimized starting point, significantly improving the sampling efficiency of the primary RL agent during its learning phase.

III. ROSE-Opt: Robust and Efficient Device Optimization with Knowledge-Infused Learning

In this section, we introduce the proposed ROSE-Opt framework that automates the P2S task. We start with the problem formulation. Then, an overview of the ROSE-Opt framework is presented, followed by an elaboration of the BO vanguard. Finally, we introduce five essential parts of the RL backbone and show how the key domain knowledge is incorporated into the framework.

A. Problem Formulation

We target the challenging device sizing problem with a given circuit under stringent PVT variations and parasitic effects of physical layout, formulated as

$$\begin{aligned} \min_{s} \quad & f(s,g), \\ \text{s.t.} \quad & s = F(x), \ \text{where} \quad s \in \mathbb{R}^{i\times j}, \ x \in S_P, \ g \in S_G. \end{aligned} \tag{1}$$

Here, the function $f(s,g)$ represents the difference between the circuit specifications s and the design goal g. For example,

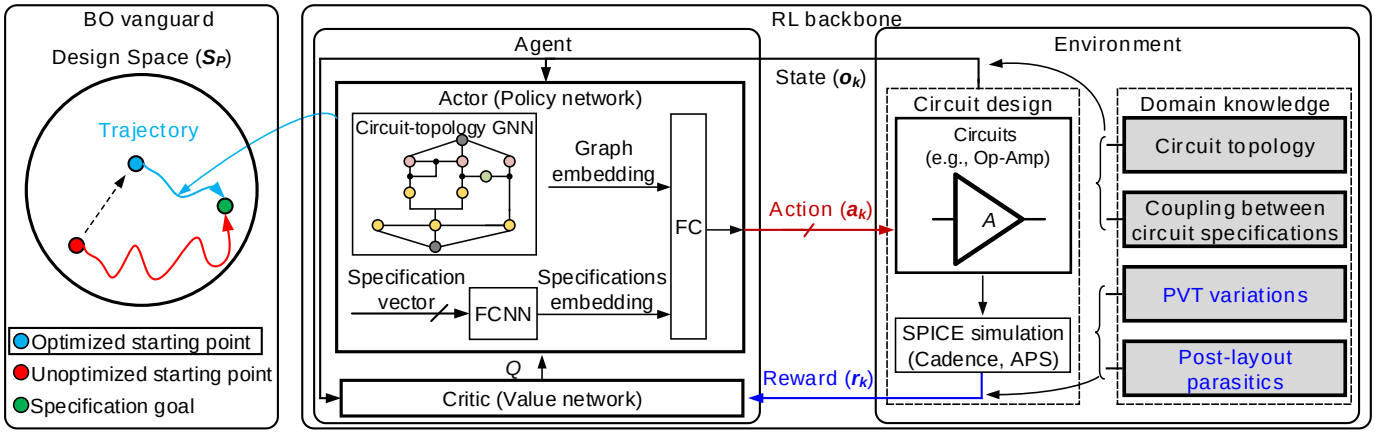

Fig. 3: Overview of our ROSE-Opt framework for automated design of analog circuits. The RL agent is based on an actor-critic method. The environment consists of a netlist of an analog circuit with a given topology, a circuit simulator, and a data processor. At each time step k, the agent automatically produces an action a_k to update device parameters with its policy network according to the state o_k and then receives the reward r_k from the environment. Our policy network is composed of a circuit topology-based GNN and an FCNN.

for operational amplifiers (Op-Amps), there are four main circuit specifications, i.e., gain (G), power consumption (P), phase margin (PM), and bandwidth (BW). $F(\cdot)$ is a circuit simulator environment to get circuit specifications s based on a set of device parameters x, e.g., width and finger numbers of transistors. s is essentially a matrix where i represents the specification type (e.g., gain) and j represents a PVT corner. Therefore, $s \in \mathbb{R}^{4\times 16}$, assuming 16 PVT corners for the design of Op-Amps. The set of device parameters x is restricted by the design space S_P. The design goal g is restricted to a reasonable sampling space S_G that the circuit can achieve. Our objective is to minimize $f(s, g)$ by efficiently looking for a group of optimal device parameters so that the circuit specifications can meet an arbitrarily given group of design goals under all PVT variations. Considerations of parasitic effects are discussed in Section V-B as it is considered during the deployment stage rather than the training stage.

B. Framework Overview

We explore the synergy of BO and RL to achieve robust and sampling-efficient device parameter optimization. Fig. 3 shows the overview of the proposed ROSE-Opt framework, which contains two parts: a BO Vanguard and an RL Backbone. BO is a well-known optimization algorithm that often achieve the fastest convergence [1] to an optimum for a given design goal, compared to other optimization techniques [13], [27]. However, it needs to be restarted from scratch if the given design goal is changed.

In contrast, well-trained RL agents can reach general design goals without retraining based on a deployment trajectory from a starting point. Unfortunately, for robust analog circuit design, which is a more complex problem, RL methods demand more data points from time-consuming circuit-level simulations (i.e., PVT simulations) to sufficiently explore the design space, leading to a low sampling efficiency toward the convergence. With this key insight in mind, we propose to leverage BO as a vanguard to first coarsely search for an optimized starting point (i.e., initial device parameters) for our RL agent. On this basis, the RL agent can then be trained with much fewer interactions with time-consuming circuit-level simulations, improving the sampling efficiency. As conceptually shown in the left subset of Fig. 3 (i.e., BO vanguard), an optimized starting point can help the RL agent reach design goals with a shorter trajectory compared to a randomly selected one. Hence, it can guide the RL agent to converge faster with fewer training data.

The RL backbone has five essential components (Section II-B): reward, action space, state space, environment, and agent. To train an excellent RL agent for a given task, there are several critical factors to pay attention to. One is to develop a comprehensive environment that could expose environmental information about the task to the RL agent as much as possible. The second is to capture sufficient exposed observations (states) relevant to the task from the environment into the learning loop. The third is to design a proper reward function that is closely related to the optimization goal and stimulates the learning of the RL agent. With these key factors in mind, we distribute the domain knowledge presented in Section II-C across different components of RL.

Comprehensive Environment. First of all, we develop a thorough circuit design environment for the P2S task, which includes the full circuit netlist of the given task and commercial simulation/verification tools (e.g., Cadence Spectre) for simulating circuit specifications (under PVT variations) and extracting post-layout parasitics.

Sufficient Observations. On the basis of the developed environment, we leverage the circuit topology and the simulated circuit specifications as primary observations. The circuit topology and couplings between circuit specifications are incorporated into the learning loop of the RL agent through a novel policy network by combining a circuit topology-based graph neural network (GNN) and a fully connected neural network (FCNN). In particular, the policy network can effectively capture the essential physical features (e.g., device parameters and interactions) embedded in a circuit graph with the GNN and extract the couplings (i.e., design trade-offs) between circuit specifications with the FCNN, which better models the relations between the circuit parameters and the design targets.

Custom Reward Function. PVT variations affect the circuit specifications, which are directly related to the optimization goal. We use a custom reward function that takes into account

PVT variations. By infusing the key domain knowledge into the ROSE-Opt framework in this manner, an excellent RL agent can be trained and make good decisions to search for reliable device parameters that meet the design goals.

For RL training, in each episode, the agent starts from an initial state o_0 with a group of initial device parameters optimized by BO vanguard and a group of randomly-sampled desired specifications g from sampling space S_G. The end of an episode is when the design goals are realized or a predefined maximum step T is reached. At each time step k, the agent begins by using a neural network to observe a state o_k and take discrete action a_k based on the probability distribution from the output of the neural network. Then, the agent arrives in a new state o_{k+1} and receives a reward r_k from the environment. The discrete action a_k can simultaneously update all device parameters for the given circuit. The agent iterates through the episode with multiple steps and accumulates the reward at each step until the end of the episode. In the next episode, the agent randomly samples another design goal g from the sampling space S_G and reset the parameter back to the starting point o_0. Then, repeat the same process again. Once the policy network is well-trained, we can save the weight of the neural network for deployment. During the deployment, since the weight has already been trained, the agent only uses the actor to take actions based on the state it observed. The purpose of the deployment part is to show the generalization capability of our trained policy network to different specifications without retraining like BO. Therefore, we are interested to see how many specifications the decision policy can reach within the predefined maximum step T and what is the average deployment length for each run.

A key point is that BO is only required once in our framework if the sampling space S_G of the design goals and the design space S_P of each circuit device are defined. The same optimized starting point o_0 is then used by the RL agent during each training episode and the deployment stage. Note that in the context of robust device sizing, the designs of both the BO vanguard and the RL backbone are non-trivial and they are elaborated in the following.

C. BO Vanguard

We rely on BO to find an optimized initial search point for our RL agent to improve its sampling efficiency during training. However, a crucial initial question is how to define such an optimized starting point. This starting point should not just speed up the design for a specific set of goals but should broadly aid in the efficient design of any arbitrary group of design goals from the entire sampling space S_P.

Our idea is that from this starting point, the RL agent should generally take the least deployment steps to achieve a general design. Thus, we let the device parameters found by BO that achieve as closely as possible the arithmetic mean of the maximum/minimum of each design goal in the entire sampling space S_G, i.e., $(PM_{\max}+PM_{\min})/2$, $(G_{\max}+G_{\min})/2$, $(BW_{\max}+BW_{\min})/2$, and $(P_{\max}+P_{\min})/2$, be the starting point of our RL agent. Here, taking a two-stage Op-Amp as an example, G, B, PM, P are the circuit specifications, i.e., gain (G), bandwidth (B), phase margin (PM), and power consumption (P).

We use a typical set-up of BO to search for the optimized starting point, which includes two essential parts: the surrogate model and the acquisition function [1]. The whole optimization depends on how accurately the surrogate model estimates the black-box function. Particularly, we adopt the widely used Gaussian process model as our surrogate model to predict the underlying black function with uncertainty. We use a Monte-Carlo-based Expected Improvement (EI) acquisition function to balance exploration and exploitation during the optimization by offering the next sampling point as below:

$$\begin{aligned} \mathrm{EI}(X) &\approx \frac{1}{Z}\sum_{i=1}^{Z} \max_{j=1,\ldots,n} \left\{\max\left(\xi_{ij} - f(s,g)_{best}, 0\right)\right\}, \\ \xi_i &\sim \mathbb{P}(f(X) \mid \mathcal{D}). \end{aligned} \tag{2}$$

Here, the expectation, $\mathrm{EI}(X)$, is computed by approximating the integrals over the posterior distribution over Z points using Monte-Carlo sampling. $\mathbb{P}(f(X) \mid \mathcal{D})$ is the posterior distribution of our function $f(s,g)$ at X where $X = (x_1, \ldots, x_n)$ from the sampling in our design space S_P. $\mathcal{D}$ is our data set. The parameter ξ determines the amount of exploration during optimization.

D. RL Backbone

The RL backbone has five essential components:

1) Variation-aware reward function: We connect the objective in Eq. (1) to our reward function so that our RL agent can be directly optimized considering PVT variations. Particularly, the reward r_k at each time step k is designed by taking PVT variations into consideration, i.e.,

$$\begin{aligned} r_k &= \mathrm{Mean}\Big(\sum_{j=0}^{j=M-1} r_j\Big);\ \text{if } \exists j \in [0,\ M-1],\ r_j < 0; \\ \text{or } r_k &= R,\ \text{if } \forall j \in [0,\ M-1],\ r_j = 0. \end{aligned} \tag{3}$$

Here, $r_j = \sum_{i=0}^{N-1} w_i \times \min\{(s_i^j - g_i)/(s_i^j + g_i), 0\}$ is the sub-reward of the j^{th} corner, calculated based on a weighted sum of the normalized difference between i^{th} intermediate circuit specification of the j^{th} corner s_i^j and i^{th} design goal g_i. All types of circuit specifications are equally important, i.e., $w_i = 1$. M represents the number of PVT corners and N indicates the number of circuit specifications. In order not to over-optimize the specification, we set the upper bound of r_j to be 0. Only when the circuit specifications in all PVT corners meet the design goal, a large stimulated reward of $R = 10$ is given to encourage the agent for the successful design; otherwise, the reward in each time step is the average of sub-rewards of all PVT corners. Finally, the accumulative reward for a training episode is $R_{s,g} = \sum_{k=1}^{T} r_k$, where T is a pre-defined maximum step for an episode. Intermediate circuit specifications matrix s are obtained from our high-fidelity simulation environment $F(\cdot)$ based on the updated device parameters x at each time step. Therefore, our reward is a direct measurement from the circuit simulator, which can help train a high-quality RL policy network.

2) Fine-grained action: Inspired by human designers who rely on multiple fine-grained tuning steps to find optimal device parameters, we use a discrete action space to tune device parameters. For each tunable parameter x of a device (e.g., the width and finger number of transistors, the capacitance of capacitors), there are three possible actions: increasing ($x + \triangle x$), keeping ($x + 0$), or decreasing ($x - \triangle x$) the parameter, where "$\triangle x$" is the smallest unit used to update the parameter within its bound $x \in [x_{\min},\ x_{\max}]$. With the total parameters of the M device, the output of the policy network is a matrix of probability distribution $M \times 3$ in any state where each row corresponds to a parameter. The action is taken based on the probability distribution.

3) Circuit physics-related state: RL belongs to representation learning. Capturing adequate state information from the environment is key to training an excellent RL agent. We leverage the domain knowledge of analog circuits, i.e., intermediate circuit specifications and circuit topology, as our state, which covers the most essential observations from a circuit design environment. In particular, we take care of intermediate circuit specifications in all PVT corners, in contrast to the previous work [28] which only considers partial PVT corners. We create a state vector to represent the intermediate circuit specifications, which is used as input for the FCNN part in our policy network. To better use the observations of the circuit itself, we use a graph $G(V,\ E)$ to model the circuit according to its topology, where each node in the set V is a device and the connections between the devices constitute the edge set E.

Fig. 4 shows the mapping between the circuit topology and a graph taking a two-stage Op-Amp as an example. For a circuit with n nodes, the state of the i^{th} node is defined as its node feature $(t,\ \vec{p})$, where t is the binary representation of the type of node and $\vec{p}$ is the parameter vector of the node. Note that the parameters of the circuit device reflect the physical information of the circuit. For transistors, the parameters are the width (x_{W}) and the finger number (x_{F}). For capacitors, resistors, or inductors, the parameters are scalar values (e.g., capacitance, resistance, or inductance) of each device. For example, for a circuit with five different types of devices, the state of a N type transistor can be expressed as $[0,\ 0,\ 1,\ x_{\text{W}},\ x_{\text{F}}]$.

4) SPICE simulation environment: In our work, a high-fidelity circuit design environment with PVT variations and post-layout parasitics is used. It consists of the netlist of a given analog circuit, a commercial circuit simulator, e.g., Cadence Spectre for CMOS analog circuits or Keysight Advanced Design System (ADS) for RF power circuits, and a data processing module (DPM). As shown in Fig. 3, the simulator obtains intermediate circuit specifications at each time step. The DPM then deals with the simulated results to return a reward to the agent using Eq. (3). Meanwhile, it also updates the device parameters to rewrite the circuit netlist based on the actions of the agent (i.e., policy network).

Previous methods [28] assume that the circuit simulation time scales linearly with the number of simulations, i.e., the simulation time for 16 PVT corners is 16× of the one with a single PVT corner. We use the Cadence Spectre Accelerated Parallel Simulator (APS) to accelerate our simulation. At each time step, we obtain circuit specifications for all PVT corners.

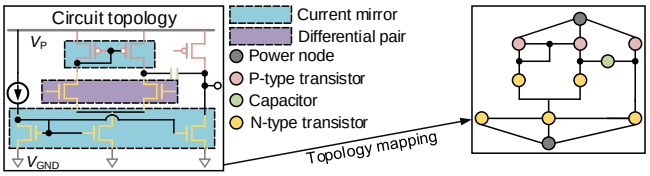

Fig. 4: Mapping a circuit topology into a graph and illustrating a tailored GNN-FC-based policy network architecture for the analog circuit design. Here, we use a two-stage Op-Amp as an example.

Compared to a single PVT corner time, the batch simulation manner for 16 PVT corners only brings 0.17× time overhead as compared to a single PVT simulation. In other words, our circuit environment can achieve a sampling efficiency of at most 14× when collecting data points during training compared to previous RL methods. With this co-design loop, we are able to simultaneously achieve both high sampling efficiency and robust design by taking advantage of BO, RL, and the simulation environment.

5) Circuit-aware policy network: We adopt an Actor-Critic method [29] to design our agent. To capture sufficient observations from the environment to the learning loop, we propose a novel multimodal policy network architecture for the Actor, as shown in Fig. 4. The policy network consists of a GNN based on the circuit topology and an FCNN, which is termed a GNN-FC-based policy network. Specifically, the GNN is used to distill the underlying physics (e.g., device types, parameters, and interactions) of a circuit graph into a low-dimensional vector embedding. The FCNN takes the design goals as inputs to extract their coupled relations, i.e., design trade-offs. The graph embedding and the FCNN embedding are then concatenated for further processing by the final FC layers to update the actions. The value network (Critic) has the same architecture as the policy network except for the last layer. It evaluates the quality of the actor's decision by giving an estimate of the expected reward, Q, for the execution of the current policy. In particular, we choose graph attention network (GAT) [30] as a representative of the GNN to model the circuit topology. The goal is to learn the embedding of physical features at the circuit level (e.g., device parameters, interactions, and types) on a circuit graph $G = (V,\ E)$. Our empirical studies show that GAT often performs better than other GNNs such as the graph convolutional network (GCN) [31] in the P2S task. This could be attributed to the multi-head attention mechanism of GAT, which helps to learn more complex and higher-dimensional interactions between a circuit node and its neighbors. Note that we customize the GAT to model circuit topologies and apply them to the P2S task rather than inventing novel GNN structures. The fundamental operations underlying the proposed GAT follow those of the original publications [30] and therefore are omitted here.

E. Optimization Methods for Policy Training

Combining the GAT and FCNN forms the policy network $\pi_\theta(a|s)$ parameterized by $\theta = \{W_{\text{GAT}}, W_{\text{FC}}\}$. Here, W_{GAT}, W_{FC} are the learnable parameters for the GAT and FCNN. Our goal is to make the RL agent gain rich circuit design experiences and generate higher-quality decisions by interacting

TABLE I: Design space, sampling space, and PVT corners for four benchmark circuits.

Circuit types	Single-stage Op-Amp	Two-stage Op-Amp	Folded-cascode Op-Amp	Nested Miller compensation Op-Amp
Technology	GlobalFoundries 130/65/28 nm			
16 PVT conditions	Process:{SS, SF, FS, FF}		Voltage:{1.1V, 1.3V}	Temperature:{$-40°C$, $125°C$}
Design space	10^{24} values	10^{10} values	10^{17} values	10^{39} values
Width (nm)	mp1-4:[200, 2000, 10] mn1-3:[2000, 10000, 10]	mp1:[1250, 2250, 10] mp2: [450, 2450, 20] mn1-2:[160, 260, 1]	mp1:[1000, 10000, 200] mp2: [1000, 10000, 200] mn1-3:[160, 1000, 20]	mp1-3&mn4:[10000, 50000, 1000] mp4: [50000, 250000, 10000] mn1-3:[2000, 20000, 1000]
Capacitance (pF)	C_L: 0.12	C_L: 1	c:[0.1, 10.0, 0.2] C_L: 1	c1:[25.0, 50.0, 0.5] c2:[1.0, 25.0, 0.5] C_L: 100
Sampling space				
Gain (dB)	$[40, 45]$	$[10, 15]$	$[20, 30]$	$[40, 45]$
I (A)	$[10^{-5}, 10^{-4}]$	$[10^{-3}, 10^{-2}]$	$[10^{-4}, 10^{-3}]$	$[10^{-2}, 10^{-1}]$
PM (∘)	[50]	[55]	[85]	[55]
BW (Hz)	$[5 \times 10^5, 1 \times 10^6]$	$[10^5, 4 \times 10^5]$	$[4 \times 10^6, 6 \times 10^6]$	$[1 \times 10^6, 2 \times 10^6]$

with the environment. We can formally define the objective function of automated design of analog circuits as follows.

$$J(\theta, G) = 1/H \cdot \sum_{g \sim G} \mathbb{E}_{g, s \sim \pi_\theta}[R_{s,g}]. \quad (4)$$

Here, H is the the space size of all desired specifications G and $R_{s,g}$ is the episode reward. Given the cumulative reward for each episode, we use Proximal Policy Optimization (PPO) [32] to update the parameters of the policy network with a clipped objective shown below:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_k[\min(b_i(\theta), \text{clip}(b_k(\theta), 1-\epsilon, 1+\epsilon))\hat{A}_k], \quad (5)$$

where $\hat{\mathbb{E}}_k$ represents the expected value at time step k; b_k is the probability ratio of the new policy and the old policy, and $\hat{A}_k$ is the estimated advantage at time step k.

Previous RL-based methods [17], [18] for P2S tasks mainly explore Deep Deterministic Policy Gradients (DDPG) to train RL agents and have also shown promising performance. However, the lack of a detailed comparison between different RL algorithms makes it difficult to determine which is better for P2S tasks. DDPG is an off-policy RL method that uses two separate policies for exploration and updates, a stochastic behavior policy for exploration, and a deterministic policy for the target update. The "deterministic" in DDPG refers to the fact that the agent computes the action directly instead of a probability distribution over actions. DDPG is specifically designed for environments with continuous action spaces and continuous state spaces, making it an equally valid choice for continuous control tasks applicable to fields such as robotics or autonomous driving.

On the other hand, PPO is an on-policy RL method, that is, it involves collecting a small batch of experiences by interacting with the environment according to the latest version of its stochastic policy and using that batch to update its decision-making policy. The "stochastic" in PPO refers to the fact that the agent computes the action as a probability distribution instead of directly over actions. PPO often can work with both discrete and continuous action spaces, making it suitable for a wide range of reinforcement learning tasks in various domains, e.g., training ChatGPT. In particular, we use RL with discrete action space to build our framework due to: 1) experienced human designers also use fine-grained tuning (i.e., adjusting device parameters with several discrete tuning steps) to tackle the P2S task; 2) the thorough study in Section V-D shows that PPO with discrete action space achieves better performance.

IV. Experimental Methodology

In this section, we present the experimental methodology for evaluating the proposed framework. First, we introduce the circuit benchmarks used in our evaluations. Then, baselines for comparisons are briefly discussed. Finally, we show the training platform and configurations of our framework.

A. Benchmarks and Performance Metrics for Evaluation

Operational amplifiers (Op-Amps) are commonly used as circuit benchmarks in prior art [15], [16], [18], [25], [28], [33] and are also widely used as essential building blocks in many analog subsystems. Therefore, we take multiple Op-Amps to evaluate the proposed framework. In particular, we adopt a single-stage cascode Op-Amp, a two-stage Op-Amp, a folded-cascode Op-Amp [34], and a three-stage nested Miller compensation Op-Amp with feedforward transconductance stage [35] (NMCF) in our benchmark. These circuits have diverse topologies and design complexities. The detailed schematics of these circuits were shown in previous work [15], [16], [18], [25], [28], [33] and are thereby omitted here. The design space for the device parameters, the sampling space for the circuit specifications, and the PVT corners are listed in Table I. There are $4 \times 2 \times 2 = 16$ extreme PVT conditions, including 4 process variations, 2 voltage variations, and 2 temperature variations.

With the circuit benchmark, we examine mainly the sampling efficiency, design success rate, and design efficiency of our framework. We show the sampling efficiency of ROSE-Opt by using a control experiment, that is, to train the RL agent with/without the BO vanguard. The sampling efficiency is defined as the number of SPICE simulations saved to achieve the same training quality (i.e., training reward) compared to the control group.

To allow a reasonable comparison between different design automation methods, we also propose a figure-of-merit (FoM) defined as the ratio between the design success rate (N_{success}) and the design efficiency ($N_{\text{step}} \cdot T_{\text{sim}}$): FoM $= N_{\text{success}}/(N_{\text{step}} \cdot T_{\text{sim}})$. Here, N_{success} is the design success rate of policy deployment by giving 200 groups of design goals

randomly sampled from the specification space. N_{step} is the average number of required deployment steps (i.e., the number of circuit-level simulations) to achieve a group of design goals sampled from the specification space. T_{sim} is the simulation time for each simulation run at the circuit level. Note that the training time for learning-based methods is not included here, as it can be amortized during the deployment phase once the models are trained well (similar to the inference stage in supervised learning) .

B. Training Platform, Configurations, and Baselines

Our framework is built on Python. We build the circuit graph using the Deep Graph Library [36] and use Ray [37], a well-developed hyperparameter tuning package, to train RL agents. We implement all the methods with PyTorch and BoTorch [38]. All experiments were carried out on a 16-core Intel CPU. We train separate RL agents for each circuit. For experiments that involve the BO vanguard to optimize the initial starting point, we only assign 50 simulations in each BO run for minimal sampling overhead. Note that we only need to run BO once at the beginning since we can reuse the starting point optimized by BO vanguard in each RL training episode and deployment stage. To achieve a more reliable and reproducible experiment result, we decided to run our BO vanguard 50 times and choose the starting point with a mean reward to minimize the variation caused by the initial random sampling. To provide detailed comparisons of the performance of different RL algorithms, we choose PPO [32] and DDPG [39] as two representatives of the study and also use their default configurations to train policy networks.

Although various previous methods have been proposed to target the P2S tasks, such as BO [1], Genetic Algorithm [40], and RL methods [18], [25], they do not consider PVT variations and post-layout parasitics in the optimization process of device parameters. Thus, we compare it with the most recent work, RobustAnalog [28], which solves the P2S task by taking into account the effect of partial variations in PVT. Despite several major differences between RobustAnalog and ROSE-Opt, we care most about the efficacy of RobustAnalog in robust design, as it uses task pruning with reduced PVT corners for RL training, while our RL Backbone considers all PVT corners. We follow this strategy to implement RobustAnalog by modifying our RL backbone.

V. Experimental Evaluations

In this section, we show evaluation results and compare the performance of our proposed framework to prior methods. First, we show our framework's sampling efficiency and robustness against PVT variations. Second, we show our framework's capability to achieve reliable device sizing by taking into account post-layout parasitics. Third, we show how the trained RL agent of our framework assists human designers in finding optimized device parameters, even if it fails in deployment in some cases. Fourth, we present the performance of different RL algorithms in training RL agents for the P2S task. Finally, we summarize the comparisons between our work and the prior art.

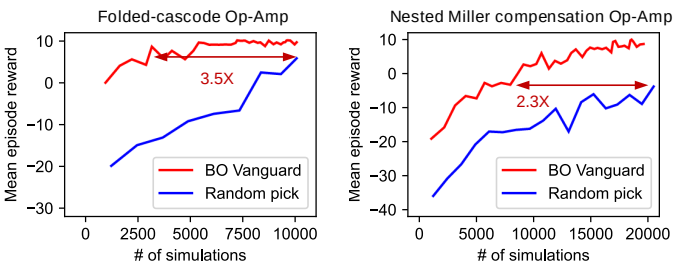

Fig. 5: An example to show comparison of the sampling efficiency by using the RL Backbone with or without pre-optimization of BO.

A. Sampling Efficiency and Robustness

1) Efficient sampling with BO Vanguard: We first show that the BO vanguard can improve sampling efficiency to train our RL backbone. Fig. 5 illustrates the example training curves of our RL backbone to design two types of Op-Amps with two different starting points, one is from BO searching (labeled as "BO Vanguard") and the other is a randomly selected value from the device parameter design space S_P, e.g., median value (labeled as "Random pick"). It shows that the RL agent without an optimized starting point often needs more circuit-level simulations to achieve the same reward as the one with an optimized starting point from BO (e.g., in this case, 3.5× for Folded-cascode Op-Amp and 2.3× for three-stage nested Miller compensation Op-Amp). Therefore, by optimizing the starting point, the RL agent converges faster with fewer sampling data (that is, fewer circuit-level simulation runs).

2) Robust design with the RL Backbone: We then show the robust design enabled by our RL backbone against PVT variations by deploying our RL agent in an environment taking full account of PVT variations. Policy deployment applies a trained policy to automatically find the device parameters for given design goals. The left column of Fig. 6 shows the deployment trajectories under several representative PVT corners by taking the phase margin of the Folded-cascode Op-Amp as an example, where each color represents a PVT corner. It can be seen that although each trajectory under a specific PVT corner is smooth, the worst corner can be quickly replaced by another corner due to the competition between different corners. Here, the worst case indicates the corner where the circuit specification deviates the most from the design goal. This phenomenon shows that device sizing with PVT variations is much more complex compared to the nominal case. Notably, by incorporating PVT variations into our method, our RL agent is able to achieve a robust design by finding optimal device parameters that can satisfy the design goal in all PVT corners.

For further verification, we conduct a control experiment. We first find a group of optimal device parameters by deploying our trained RL agent at the nominal corner environment. We then verify the circuit specifications under all other PVT corners using the found device parameters. The right column of Fig. 6 shows that the optimal device parameters obtained from a single PVT corner often do not satisfy all corners.

TABLE II: Comparison of device parameters before and after physical design with design goals of gain = 45 dB, bandwidth = 800 kHz, phase margin = 50°, and power = 15 μW.

Device	mp1	mp2	mp3	mp4	mn1	mn2	mn3	Gain	BW	PM	Power
Schematic design	1.02 μm	1.02 μm	1.02 μm	1.02 μm	6.2 μm	4.94 μm	6.2 μm	47.3 dB	862 kHz	52°	12 μW
Layout design	1.02 μm	1.02 μm	1.02 μm	1.02 μm	5.26 μm	5.26 μm	5.26 μm	46.91 dB	897 kHz	53.52°	12 μW

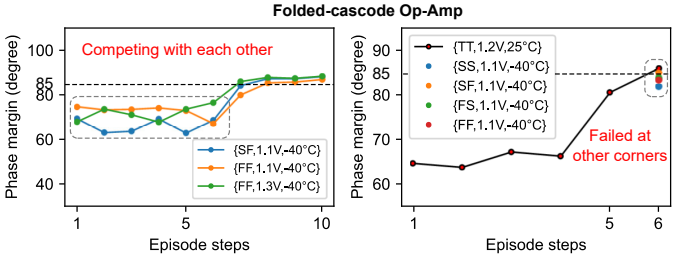

Fig. 6: Left: illustration of the competing phenomenon between PVT corners. Right: failed design by using the deployment with a single PVT corner at the nominal case. The dashed horizontal line is the design goal.

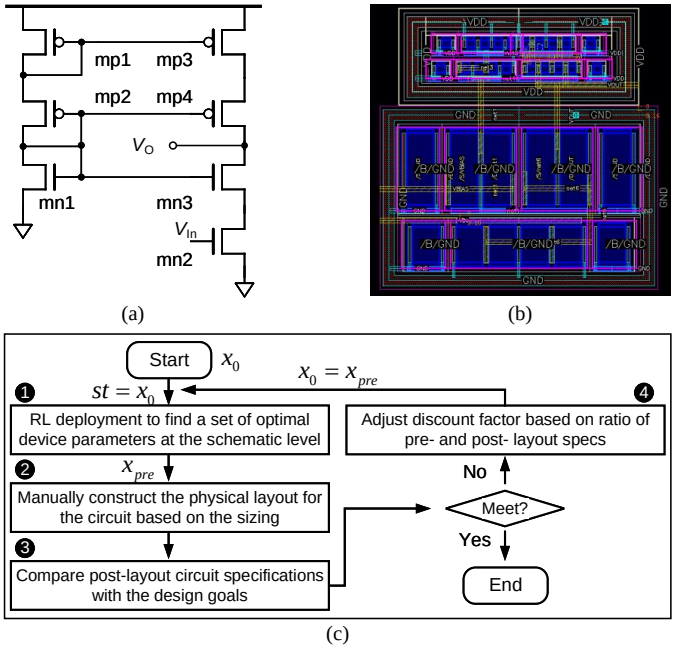

Fig. 7: Single-stage Op-Amp used for parasitic-aware sizing. The figure shows (a) its schematic, (b) its physical layout, and (c) the flow to use ROSE-Opt for the parasitic-aware sizing.

B. Parasitic-Aware Device Parameter Optimization

We continue to study how to apply ROSE-Opt to optimize parasitic-aware device parameters. Without considering the parasitic effect of physical layouts at the pre-layout design stage, the obtained device parameters are not able to guarantee the circuit specifications after the schematic is directly transferred into a physical design. In practice, there are often tens of iterations between schematic design and physical design performed by human designers to fine-tune the device parameters to ensure that the circuit under design meets the design goals.

Several previous works have explored learning-based methods to address this parasitic-aware optimization problem. An early RL-based method [25] aims to tackle it by deploying the trained RL agent in a parasitic-aware environment. In particular, this method uses the BAG tool [10] to automatically generate a physical layout for the circuit based on the device parameters at each deployment step and exploits the reward from post-layout simulation to guide the search process for the trained RL agent. This process continues until the agent meets the target when parasitics are considered or it has reached the maximumly-allocated deployment steps. Another work [41] attempts to tackle the problem by combining supervised learning and BO. The idea is to train a graph neural network to predict the parasitics of an analog circuit with its device parameters and then back-annotate the parasitics to the circuit schematic. With this processing step, BO is applied to search for optimal device parameters through the parasitic-aware schematic.

These previous efforts have shown good performance in finding reliable device parameters to meet design goals after post-layout simulation. However, the physical design of analog circuits is quite flexible. Even for the same circuit, different human designers can construct different physical layouts. The BAG tool is limited to generating a few fixed layouts for some typical circuits. Training a GNN to predict parasitics requires a huge amount of data and suffers from approximation error. Therefore, the previous methods do not apply to general cases.

We explore another method to solve the same problem with much higher flexibility. Our method relies on two key observations from the human design loop. First, human experts often construct an initial physical layout of the circuit with an initial set of device parameters and fine-tune the device parameters by following the same placement of the device as the one used in the initial physical layout. Second, circuit specifications from the post-layout simulation of this initial physical layout are often degraded compared to the desired goals but are not far from them. Therefore, the optimal final device parameters to meet the design goals also fall in the neighborhood of the initial set of device parameters.

With these key observations in mind, our method can apply to parasitic-aware device parameter optimization by following the essential steps shown in Fig. 7(c). We begin by initializing all discount factors to 1. These discount factors are explained in ❹. The other steps are as follows:

❶ Deploy the trained RL agent to find the set of optimal device parameters that satisfy the design goals in the pre-layout stage and perform simulations to obtain the circuit specifications with this set of device parameters; the i^{th} specification of the j^{th} corner is marked as $s^j_{i,\text{pre}}$.

❷ Construct a physical layout with the device parameters found in Step ❶.

❸ Extract the circuit specifications (i.e., $s^j_{i,\text{post}}$) of this physical layout by performing a post-layout simulation and compare them with the design goals; if satisfied, the design is successful; otherwise, jump to Step ❹.

❹ Adjust the discount factor based on the ratio between pre- and post- layout specifications, e.g., if $s^j_{i,\text{pre}} \geq s^j_{i,\text{post}}$, $\alpha^j_i = s^j_{i,\text{post}}/s^j_{i,\text{pre}}$.

TABLE III: Detailed device parameters during the policy deployment for the two-stage Op-Amp.

Device parameters	Width of mp1 (μm)	Width of mn1 (μm)	Width of mp3 (μm)	Width of mn3 (μm)	Width of mn4 (μm)	Width of mn5 (μm)	Capacitance of c1 (pF)	Reward
Step 26	11	35	79	18	6	46	4.4	−0.085
Step 27	11	35	81	17	5	45	4.4	−0.105
Step 28	10	34	83	16	4	47	4.3	−0.057
Step 29	10	34	85	15	4	49	4.5	−0.071
Step 30	10	33	85	14	3	48	4.5	−0.088
Manual tuning	10	35	83	17	4	47	3.6	10

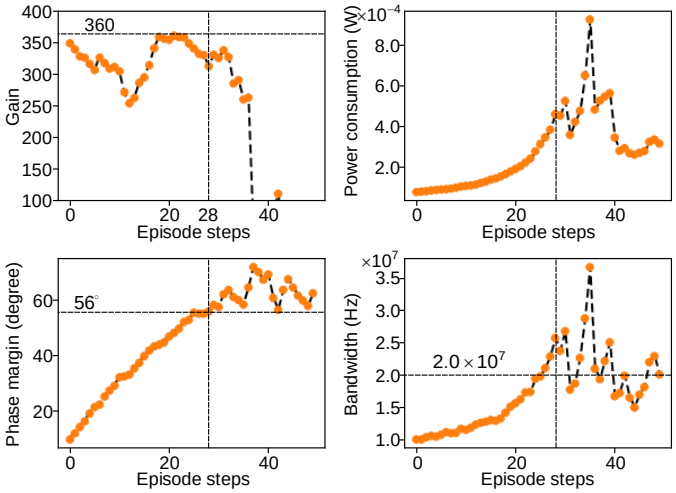

Fig. 8: Failed policy deployment in the two-stage Op-Amp example. The highest reward appears in the 28th step. After slight manual adjustment from that step, a set of optimal device parameters can often be easily obtained as shown in Table III.

After the first iteration, we repeat the flow using the set of device parameters x_{pre} found in Step ❶ as the new starting point st of the trained RL agent for another deployment and the intermediate circuit specification in ❶ will be discounted by the discount factor as $\alpha_i^j \cdot s_{i,\text{pre}}^j$, until the optimal final device parameters that satisfy the design goals in the pre-layout stage are found. Our experiments show that usually it takes no more than two rounds to reach a set of device parameters with which the circuit specifications of a physical layout can also meet the design goal. Table II shows optimized device parameters with/without consideration of parasitic effects, and Fig. 7(b) shows the final physical layout corresponding to the reliable device parameters for the circuit shown in Fig. 7(a).

C. Analysis of Failed Deployment Cases

Our trained RL agent achieves a high design success rate with policy deployment (i.e., >90% across different circuits as reported by our prior work [19]). Despite great promise, we analyze a few failed cases where our trained policy cannot converge to the optimal device parameters. We find that for these failed cases, some circuit specifications are able to reach the design goals, while the others converge to a neighborhood of the desired ones at some deployment steps, but after which they deviate a bit from the goals.

Fig. 8 shows such a failed policy deployment in the two-stage Op-Amp example, where the desired circuit specifications given are gain ($G = 360$), bandwidth ($B = 2.0 \cdot 10^7$ Hz), phase margin ($PM = 56°$), power consumption ($P = 6.93 \cdot 10^{-3}$ W). It is observe that around the 28th step, the bandwidth, phase margin, and power consumption are satisfied, but the gain is still lower than the design goal. We examine the detailed device parameters[1] around the 28th step as shown in Table III. It shows that the reward achieves the highest value in the 28th step. We then select the device parameter values reached at this step and proceed with a slight manual tuning starting from these values. With less than five manual tuning iterations, a set of optimal device design parameters can often be easily obtained. The last row of Table III shows the device parameters obtained after slight manual adjustment.

D. Comparisons between Different RL Algorithms

Different RL algorithms have shown different performance in solving practical problems. PPO and DDPG are two primary RL algorithms used in current RL-based methods [17], [18], [25] for P2S tasks. Here, we perform detailed experiments to compare the performance of DDPG and PPO in tackling the P2S task by using the design of a two-stage Op-Amp as an example. Fig. 9 illustrates our evaluation results, where we train RL agents with PPO using both "discrete" and "continuous" actions, as well as DDPG with "continuous" action. Each curve in Fig. 9 is based on 6 random seeds. Note that for other types of Op-Amps, we also observe similar results.

1) PPO-continuous vs. DDPG-continuous: Compared to DDPG-continuous, we find that PPO-continuous has a lower sampling efficiency during the training process. This is because PPO-continuous adopts an on-policy learning mechanism that samples actions according to its latest stochastic policy. The on-policy characteristic introduces variance, since each estimate of an expectation over a finite set of samples may vary, which necessitates a large number of samples for accurate mean calculations, thereby leading to low sampling efficiency. In contrast, DDPG-continuous utilizes an off-policy learning mechanism, which involves a replay buffer to store transitions from the previous policy and relies on the current policy only to replenish the buffer, improving the sampling efficiency. The lower sampling efficiency of PPO-continuous also impacts the training quality of the policy. As shown in Fig. 9, with the same number of samples, the episode length (deployment accuracy) of the trained policy with PPO-continuous (green line) is longer (lower) than that of the one with DDPG-continuous (red line).

[1]Note that here we do not show the finger numbers of transistors, because they generally remain unchanged around the 28th step.

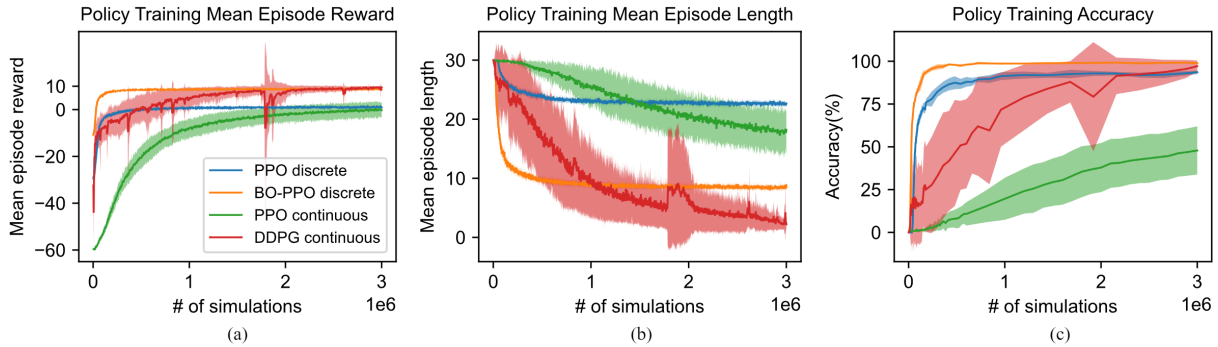

Fig. 9: Comparisons between DDPG and PPO when applied to train RL agents. (a) Mean episode reward, (b) mean episode length, and (c) deployment accuracy. Each curve in the figure is based on 6 random seeds.

2) PPO-discrete vs. DDPG-continuous: With the setting of discrete action space, PPO-discrete demonstrates superior sampling efficiency and more consistent results compared to its continuous counterpart and DDPG-continuous. In a discrete environment, the action choices of an RL agent at each step are simplified to adjust the parameter upward/downward with a small increment/decrement, or to maintain its current value. This simplicity makes the training process smoother and improves the quality of the policy to find the optimal device parameters during the deployment process, as shown in Fig. 9 (blue line). In a complex continuous environment where the agent has a plethora of parameter choices, the off-policy mechanism of DDPG-continuous could suffer from biases, where some of the updates are based on prior (potentially incorrect) expectation estimates. This leads to irreversible incorrect estimates in the end and causes training instability due to its inherent low-variance but high-bias nature. As shown in Fig. 9 (red line), there is a sudden change with respect to the mean episode reward/length and deployment accuracy.

However, there is a caveat to using PPO-discrete: the discrete environment constrains the policy's design efficiency. Its mean episode length is influenced by the granularity of the step in the discrete space and the distance between the initial state and the target solution. Thus, a longer episode length (i.e., more design steps) is required to find the optimal device parameters. In a continuous environment, the action at each step corresponds to the normalized device parameters in the design space, thereby demanding fewer design steps.

3) BO-PPO-discrete vs. PPO-discrete: To address the issue of low design efficiency of PPO-discrete, we introduce BO to optimize the starting point of PPO-discrete, thus positioning the initial state closer to the solution space. As illustrated in Fig. 9 (yellow line), this strategy could optimize the starting point, allowing PPO-discrete to reach the desired specifications without too much meaningless exploration, accelerating the trajectory formulation towards the solution. Ultimately, this integration of BO with PPO-discrete, termed BO-PPO discrete, demonstrates superior performance in both design accuracy and sampling efficiency, and achieves commendable results in design efficiency. This novel approach showcases the potential of combining classical optimization techniques to improve the effectiveness of RL in complex analog circuit design. As optimization techniques continue to evolve, the combined strengths of different optimization methods, such as BO and RL, underscore the importance of hybrid strategies in complex optimization scenarios of analog circuit parameters.

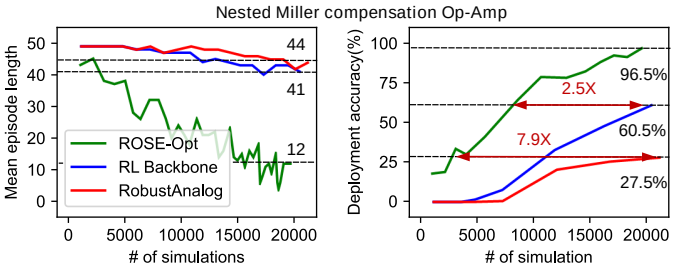

Fig. 10: Comparisons between our proposed framework ROSE-Opt, our RL Backbone, and the RobustAnalog by taking the design of Nested Miller compensation Op-Amp as an example. Right column: episode length. Right column: deployment accuracy.

4) Training Reward Is Not Always A Good Metric for Comparing Different RL Policies: One last thing to note is that DDPG-continuous is able to achieve a much larger episode reward compared to PPO-discrete during the training process but suffers from worse design accuracy during the deployment process. This discrepancy is due to the nature of the episode reward function, which accumulates intermediate rewards throughout the search process for each episode. PPO-discrete leverages fine-grained action to search for optimal device parameters through multiple steps. Due to this fine-grained mechanism, the improvement of intermediate rewards in an episode is slow, resulting in a smaller episode return. On the contrary, DDPG-continuous adopts continuous action, allowing it to find a suboptimal solution earlier in the search process and even within a single step, thereby leading to a larger accumulated reward. However, since it suffers from biases as discussed above, DDPG-continuous does not achieve a high design success rate in the deployment stage. This finding shows that leveraging the training reward as a metric to compare design automation methods based on different RL algorithms, as done in many previous work [17], [18], could be misleading. We recommend employing deployment accuracy and design efficiency as metrics for fair and reasonable comparisons across various methods.

E. Comparison and Summarization

Finally, we compare our proposed ROSE-Opt framework with a recent work, RobustAnalog [28], which also targets the optimization of variation-sensitive device parameters. Fig. 10

TABLE IV: Summary of Comparison with Existing Optimization Methods.

Methods	PVT incorporation	Optimized starting point	Sampling efficiency for policy training		FoM for policy deployment	
			folded-cascode	NMCF	folded-cascode	NMCF
RobustAnalog[a] [28]	Partial	No	1×	1×	1.14	0.625
BO Vangard[a]	Full	N/A[c]	N/A[c]	N/A[c]	0.34	0.1
RL Backbone[a]	Full	No	2×	2.5×	1.42	1.26
ROSE-Opt[a]	**Full**	**Yes**	**12×**	**7.9×**	**20.51**	**6.87**
Bayesian Optimization[b] [1]	No	N/A[c]	N/A[c]	N/A[c]	0	0
RL baseline[b] [17]	No	No	Fail	Fail	0	0

[a] RobustAnalog, BO Vanguard, RL Backbone, and RoSE consider PVT variations corners during training/optimization.
[b] Representatives of the prior arts [1], [13], [15]–[18], [25], [25], [27], [33] that ignore the variations of PVT in training/optimization. They are implemented by our BO and RL without considering PVT variations.
[c] BO methods do not need pre-optimized starting points and training, thus some metrics are not applicable here.

shows the results of the two frameworks, as well as our RL Backbone. Both ROSE-Opt and RL Backbone use full PVT corner incorporation, while RobustAnalog uses K-means clustering to select partial PVT corners during training. Moreover, ROSE-Opt uses an optimized starting point using BO, while both RL Backbone and RobustAnalog use the same randomly selected starting point. Compared to RL Backbone, RobustAnalog is less competitive by only taking partial PVT variations in the learning framework. K-means clustering has trouble clustering data where clusters have different sizes and densities. In practice, we cannot assume the cluster's shape and density based on the specifications we get from different PVT corners. Our experiment result shows that this uncertainty causes sampling efficiency issues because the RL algorithm needs more training steps to repeat the clustering whenever the device parameters reach the design goal under partial PVT corners but fail at the full PVT corners setup. On the other hand, RL Backbone's device parameters are always evaluated under full PVT corners without having this uncertainty issue. Beyond that, our ROSE-Opt framework achieves the best sampling efficiency, design efficiency, and design success rate by considering all PVT variations in the learning loop and adopting the two-level optimization method at the same time.

We summarize the features of ROSE-Opt together with previous comparisons in Table IV. Here, more works are compared against, such as directly applying prior arts [1], [13], [15]–[18], [25], [27], [33] which ignore PVT variations to perform device sizing in an environment with PVT variations. In particular, we use BO [1] and RL [25] as representatives of the prior arts, that is, optimization-based methods [1], [13], [27] and learning-based methods [15]–[18], [25], [33]. Additionally, we use the FoM defined previously to evaluate the overall performance of a design automation method in robust device sizing. Without considering PVT variations, the previous methods [1], [13], [15], [16], [18], [25], [27], [33] fail in robust design with zero design accuracy. The comparisons show that with BO as pre-optimization, our ROSE-Opt framework can significantly improve sampling efficiency, design efficiency, and design accuracy for the PVT-aware design. Note that BO is performed only once at the very beginning, thereby incurring minimal overhead. In summary, our proposed ROSE-Opt framework that takes advantage of the complementary benefits of key domain knowledge and optimization algorithms (i.e., combining BO and RL) can achieve the best FoM for the challenging reliable device sizing problem.

VI. Conclusion

We propose a RL-based framework to automate the P2S task for analog circuit design. The key property of our framework is to incorporate domain knowledge of practical analog circuit design (e.g., the underlying physical topology of a given circuit, the trade-offs between specifications, PVT variations, and parasitic effects of physical layout) into the learning loop. We show that such a framework is superior in designing various analog circuits with higher accuracy, efficiency, and reliability. We expect that our method would assist human designers to accelerate the analog chip design with artificial agents that master massive circuitry optimization experiences via learning.

References

[1] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Batch Bayesian Optimization via Multi-objective Acquisition Ensemble for Automated Analog Circuit Design," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 10–15 Jul 2018, pp. 3306–3314.

[2] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi, J. Pak, A. Tong, K. Srinivasa, W. Hang, E. Tuncer, Q. V. Le, J. Laudon, R. Ho, R. Carpenter, and J. Dean, "A graph placement methodology for fast chip design," *Nature*, vol. 594, no. 7862, pp. 207–212, Jun 2021.

[3] B. Khailany, H. Ren, S. Dai, S. Godil, B. Keller, R. Kirby, A. Klinefelter, R. Venkatesan, Y. Zhang, B. Catanzaro, and W. J. Dally, "Accelerating Chip Design With Machine Learning," *IEEE Micro*, vol. 40, no. 6, pp. 23–32, 2020.

[4] B. Razavi, *RF Microelectronics (2nd Edition) (Prentice Hall Communications Engineering and Emerging Technologies Series)*, 2nd ed. USA: Prentice Hall Press, 2011.

[5] S. Datta, "Ten nanometre CMOS logic technology," *Nat. Electron.*, vol. 1, no. 9, pp. 500–501, Sep 2018.

[6] W. Cao, X. He, A. Chakrabarti, and X. Zhang, "NeuADC: Neural Network-Inspired RRAM-Based Synthesizable Analog-to-Digital Conversion with Reconfigurable Quantization Support," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2019, pp. 1477–1482.

[7] W. Cao, L. Ke, A. Chakrabarti, and X. Zhang, "Neural Network-Inspired Analog-to-Digital Conversion to Achieve Super-Resolution with Low-Precision RRAM Devices," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1–7.

[8] W. Cao, Y. Zhao, A. Boloor, Y. Han, X. Zhang, and L. Jiang, "Neural-PIM: Efficient Processing-In-Memory with Neural Approximation of Peripherals," *IEEE Transactions on Computers*, pp. 1–1, 2021.

[9] G. Zhang, H. He, and D. Katabi, "Circuit-GNN: Graph Neural Networks for Distributed Circuit Design," in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 7364–7373.

[10] J. Crossley, A. Puggelli, H.-P. Le, B. Yang, R. Nancollas, K. Jung, L. Kong, N. Narevsky, Y. Lu, N. Sutardja, E. J. An, A. L. Sangiovanni-Vincentelli, and E. Alon, "Bag: A designer-oriented integrated framework for the development of ams circuit generators," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013, pp. 74–81.

[11] M. Degrauwe, O. Nys, E. Dijkstra, J. Rijmenants, S. Bitz, B. Goffart, E. Vittoz, S. Cserveny, C. Meixenberger, G. van der Stappen, and H. Oguey, "Idac: an interactive design tool for analog cmos circuits," *IEEE Journal of Solid-State Circuits*, vol. 22, no. 6, pp. 1106–1116, 1987.

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

[13] B. Liu, Y. Wang, Z. Yu, L. Liu, M. Li, Z. Wang, J. Lu, and F. V. Fernández, "Analog circuit optimization system based on hybrid evolutionary algorithms," *Integration*, vol. 42, no. 2, pp. 137–148, 2009.

[14] G. Wolfe and R. Vemuri, "Extraction and Use of Neural Network Models in Automated Synthesis of Operational Amplifiers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 2, pp. 198–212, 2003.

[15] Y. Li, Y. Wang, Y. Li, R. Zhou, and Z. Lin, "An Artificial Neural Network Assisted Optimization System for Analog Design Space Exploration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2640–2653, 2020.

[16] H. M.V. and B. P. Harish, "Artificial Neural Network Model for Design Optimization of 2-stage Op-amp," in *2020 24th International Symposium on VLSI Design and Test (VDAT)*, 2020, pp. 1–5.

[17] Z. Zhao and L. Zhang, "Deep reinforcement learning for analog circuit sizing," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[18] 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 *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference*, 2020.

[19] J. Gao, W. Cao, and X. Zhang, "RoSE: Robust analog circuit parameter optimization with sampling-efficient reinforcement learning," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–6.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.

[21] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3389–3396.

[22] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Young Choi, "Action-decision networks for visual tracking with deep reinforcement learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[23] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing [review article]," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[24] A. Mirhoseini, A. Goldie, M. Yazgan, J. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, S. Bae, A. Nazi, J. Pak, A. Tong, K. Srinivasa, W. Hang, E. Tuncer, A. Babu, Q. V. Le, J. Laudon, R. Ho, R. Carpenter, and J. Dean, "Chip Placement with Deep Reinforcement Learning," 2020.

[25] 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, 2022.

[26] D. M. Colleran, C. Portmann, A. Hassibi, C. Crusius, S. S. Mohan, S. Boyd, T. H. Lee, and M. del Mar Hershenson, "Optimization of Phase-Locked Loop Circuits via Geometric Programming," in *Proceedings of the IEEE 2003 Custom Integrated Circuits Conference, 2003.*, 2003, pp. 377–380.

[27] M. Hershenson, S. Boyd, and T. Lee, "Optimal design of a CMOS op-amp via geometric programming," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 1, pp. 1–21, 2001.

[28] W. Shi, H. Wang, J. Gu, M. Liu, D. Z. Pan, S. Han, and N. Sun, "RobustAnalog: Fast Variation-Aware Analog Circuit Design Via Multi-Task RL," in *Proceedings of the 2022 ACM/IEEE Workshop on Machine Learning for CAD*, ser. MLCAD '22, 2022, p. 35–41.

[29] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 1928–1937.

[30] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *International Conference on Learning Representations*, 2018.

[31] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *International Conference on Learning Representations*, 2017.

[32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 2017.

[33] Y. Li, Y. Lin, M. Madhusudan, A. Sharma, S. Sapatnekar, R. Harjani, and J. Hu, "A Circuit Attention Network-Based Actor-Critic Learning Approach to Robust Analog Transistor Sizing," in *2021 ACM/IEEE 3rd Workshop on Machine Learning for CAD (MLCAD)*, 2021, pp. 1–6.

[34] B. Razavi, *Design of Analog CMOS Integrated Circuits*, 1st ed. USA: McGraw-Hill, Inc., 2000.

[35] K. N. Leung and P. Mok, "Analysis of multistage amplifier-frequency compensation," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 48, no. 9, pp. 1041–1056, 2001.

[36] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang, "Deep graph library: A graph-centric, highly-performant package for graph neural networks," 2020.

[37] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, and I. Stoica, "Ray: A distributed framework for emerging ai applications," in *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'18. USA: USENIX Association, 2018, p. 561–577.

[38] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, "BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization," in *Advances in Neural Information Processing Systems 33*, 2020.

[39] 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," 2015.

[40] B. Liu, Y. Wang, Z. Yu, L. Liu, M. Li, Z. Wang, J. Lu, and F. V. Fernández, "Analog Circuit Optimization System Based on Hybrid Evolutionary Algorithms," *Integration*, vol. 42, no. 2, pp. 137 – 148, 2009.

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