IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 June 2024, accepted 23 July 2024, date of publication 29 July 2024, date of current version 7 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3435331

RESEARCH ARTICLE

Analog Circuit Design Automation via Sequential RL Agents and Gm/ID Methodology

SUNGWEON HONG[1], (Graduate Student Member, IEEE), YUNSEOB TAE[1], (Graduate Student Member, IEEE), DONGJUN LEE[1], (Graduate Student Member, IEEE), GIJIN PARK[2], (Graduate Student Member, IEEE), JAEMYUNG LIM[1], (Member, IEEE), KYUNGJUN CHO[3], (Member, IEEE), CHUNSEOK JEONG[3], (Member, IEEE), MYEONG-JAE PARK[3], (Member, IEEE), SONGNAM HONG[1], (Member, IEEE), AND JAEDUK HAN[1], (Member, IEEE)

[1]Department of Electronic Engineering, Hanyang University, Seoul 04763, South Korea
[2]Analog and Mixed-Signal Center, Texas A&M University, College Station, TX 77843, USA
[3]SK hynix Inc., Icheon 17336, South Korea

Corresponding authors: Songnam Hong (snhong@hanyang.ac.kr) and Jaeduk Han (jdhan@hanyang.ac.kr)

This work was supported in part by SK hynix Inc., and in part by the IITP Grant funded by Korea Government (MSIT) under Grant 2021-0-00754 and Grant IITP-2024-RS-2023-00253914.

ABSTRACT This paper studies the problem of designing analog circuits to achieve target specifications, which can be formulated as a multi-objective combinatorial optimization (MOCO) under uncertainty. We address this challenging problem using the g_m/I_D methodology and a reinforcement learning (RL) framework. The proposed fast RL-based analog circuit designer (fRL-AD) maintains circuits' DC bias conditions while determining their sizing parameters associated with AC characteristics. This ensures robust convergence to optimal sizing parameters across target specifications and proficiently captures layout effects. Specifically, by decomposing the problem into a sequence of feasible problems, our pre-trained RL agent can efficiently seek a solution for each feasible problem by generating states (i.e., candidate solutions) following a learned policy. Since the sequence of feasible regions is designed to approach an optimal solution to our main problem, the RL agent can find a near-optimal solution by sequentially tackling the feasible problems. Remarkably, using better initial points (or states), our approach is more efficient than directly solving the last feasible problem. Furthermore, we introduce an adaptive action space in our RL framework, which can dynamically modulate the size of the action space elements. The proposed method provides an effective and stable design of various analog circuits, overcoming their traditionally low productivity due to reliance on human expertise and time-consuming simulations to handle uncertainties. We verify the effectiveness of our algorithm via experiments with various analog circuit topologies.

INDEX TERMS Analog circuits, automation of analog design, combinatorial optimization, g_m/I_D methodology, reinforcement learning.

I. INTRODUCTION

Recently in the context of rapid scaling of the semiconductor manufacturing process, designing circuit systems has become increasingly complicated and demanding. A significant amount of time is required for a modern circuit to be designed, which is eventually an arduous process left for human designers to iterate through a large-scale parameter space. This issue becomes more serious for analog circuits, where designers largely rely on their intuitions to find the adequate combination of parameters to design a desired circuit. Therefore, circuit design automation methods with simulation efficiency and reliability that identify time-consuming tasks are crucial for time reduction.

Prior works for analog circuit design automation can be classified into four categories: i) knowledge-based approach,

The associate editor coordinating the review of this manuscript and approving it for publication was Jagadheswaran Rajendran.

ii) equation-based approach, iii) simulation-based approach, and iv) learning-based approach. First, in the knowledge-based approaches, circuit designers incorporate circuit equations into computer programs. [1] directly embedded manual circuit design processes in a Python-code framework. The direct codification of the manual process enabled analog circuit designers to ease the creation of analog generators. However, the knowledge-based approaches generally require a closed-form solution for a given target. To meet the given target using prior knowledge, an inaccuracy occurs by oversimplifying the given structure. Recently, to enhance the effectiveness and accuracy of the knowledge-based approaches, a new design methodology that uses a look-up table to determine the model parameters has been proposed [2].

Equation-based approaches exploit the searching efficiency through the design space of geometric programming while the constraints are given either manually or automatically [3], [4], [5], [6]. Formulating designing problems into optimization problems can provide an efficient solver under a given set of characterized constraints.

Simulation-based approaches consider circuits as black-box functions, where the circuit's performance comes from simulation to solve a combinatorial optimization (CO) problem [7], [8], [9], [10], [11], [12]. Thus, general-purpose meta-heuristic optimization methods such as non-dominated sorting genetic algorithm (NSGA-II) [7] and multi-objective evolutionary algorithm based on decomposition (MOEA/D) [8] have been proposed. Recently, stochastic evolutionary models such as Gaussian process (GP) and Bayesian optimization (BO) were considered for analog circuit automation frameworks [9], [10], [11], [12].

Learning-based approaches use machine learning as a solver for circuit designing problems. Machine learning methods include supervised, self-supervised, and reinforcement learning (RL) algorithms to determine the function between the target design specification and the output of certain parameter combinations [13]. Recently, a supervised version of graph neural networks (GNNs) was used to encapsulate the structures of nodes or topology [14], [15], [16]. RL algorithms were used in MOSFET level settings to train given topologies and learn how to take action in analog designing processes [14], [17]. Furthermore, [17] developed an RL framework that can tackle the new design specifications without retraining. Using transfer learning technology, this framework can take environmental uncertainty into account.

The former two categories require background knowledge and the physics of circuit designing to automatically design a desired analog circuit. However, the latter two categories do not require backgrounds but only employ data from the circuit simulators to evolve through several iterations. In principle, the parameter optimization of analog circuits can be formulated as CO problems. Recently, there has been a surge of scholarly contributions to propose learning-based solutions that address CO problems efficiently [18], [19]. Notably, RL frameworks to solve the CO problems have been proposed, particularly tackling renowned problems such as the traveling salesman problem (TSP), the bin packing problem (BPP), and so on. Although the proposed methods show outstanding performances over the aforementioned popular problems, the RL agent should be retrained whenever facing a new problem. This limitation prevents the solutions from being used on domain-specific CO problems. For example, heavy retraining should be performed whenever an analog circuit's target is slightly changed. Therefore, an efficient RL framework must be developed to tackle non-general and domain-specific problems without retraining. We contribute to this subject.

In this paper, we study the problem of designing an analog circuit to achieve a target specification. This problem can be formulated as a multi-objective CO (MOCO) problem under uncertainty. We overcome the uncertainty via the g_m/I_D methodology and then tackle the complex MOCO problem utilizing an RL framework. The key idea is to decompose the MOCO problem into a sequence of subproblems, each of which is nothing but a feasible problem. This decomposition is motivated by the fact that an RL framework can tackle episodic tasks (with terminal states) more efficiently than continuing tasks [20]. Note that in the context of the RL, feasible problems can be formulated as episodic tasks, whereas maximizations can be continuing tasks. Our pre-trained RL agent can efficiently seek a feasible solution by generating states (i.e., candidate solutions) following a learned policy. By construction, any terminal state is a feasible solution. Since the sequence of feasible regions is designed to approach an optimal solution to our main problem, the RL agent can find a near-optimal one by tackling the feasible problems sequentially. Noticeably, using better initial points (or states), our sequential framework is more efficient than directly solving the last feasible problem. The proposed method is named **f**ast **R**einforcement **L**earning **A**nalog circuit **D**esigner (fRL-AD).

The key features of our fRL-AD include

- Leveraging the key concept of g_m/I_D methodology, fRL-AD can optimize the parameters without disturbing the bias conditions of transistors (i.e., uncertainty) by fixing its DC operating point throughout the process. The fRL-AD optimizes the AC parameters while maintaining fixed DC operating points. This feature stabilizes the output performance of the designed analog circuit by remaining at the linear region of the operating point. Consequently, the output produced by the RL agent tends to exhibit enhanced stability and accuracy compared with traditional methods [14], [17].
- Our RL agent can solve the subframes faster by encouraging the agent to take bolder action, which we define as *adaptive action space* rather than the fixed action space. Circuit designers can also use the RL agent trained in the fixed action experiment for validation. We numerically demonstrate that the training and validation speeds become faster when the proposed

adaptive action space is adopted. In addition, the speed and stability can be balanced by choosing an adequate adaption parameter α.

- The advantage of fRL-AD is to handle device parasitics without requiring supplemental machine learning methods. This expands the scope of a single agent's coverage. A solitary agent is trained and subsequently validated across diverse environments with device parasitics. Empirical evidence demonstrates the robustness of our RL agent in processing device parasitic variations. For instance, it is observed that the trained agent adapts to consume more power to drive the same load capacitance in the parasitic device's presence. Compared with the existing machine-learning-based design approach that utilizes the g_m/I_D methodology [21], the proposed method preserves the circuit's DC operating points for robustness and captures the normalized process-and-layout-dependent parasitic parameters.

The remaining part of this paper is organized as follows. Section II provides a brief explanation of RL and the g_m/I_D methodology. We formally define our problem in Section III. The proposed method is explained in Section IV. Experimental results with various circuit topologies are provided in Section V. Some concluding remarks are provided in Section VI.

Notations. Column vectors are represented as lowercase boldface letters, and sets are presented as calligraphic capital letters. Let $\left[\mathbf{x}^\mathsf{T}, \mathbf{y}^\mathsf{T}\right]^\mathsf{T} \in \mathbb{R}^{m+n}$ be the vector concatenation between two vectors $\mathbf{x}^\mathsf{T} \in \mathbb{R}^m$ and $\mathbf{y}^\mathsf{T} \in \mathbb{R}^n$. n-dimensional interval can be represented as $(\mathbf{a}, \mathbf{b})$, where $a_i \leq b_i \ \forall i \in \{1, 2, \cdots N\}$, and $\mathbf{a} = [a_1, \cdots, a_N]^\mathsf{T}$. Given a function or a set operator f, $R(f)$ and $D(f)$ represent the range and domain of f, respectively. We denote the set of all negative real numbers as $\mathbb{R}_- = \{x \in \mathbb{R}; x \leq 0\}$. A set of positive integers from one to K is denoted as $\mathbb{Z}_K$ (i.e., $\mathbb{Z}_K = \{1, 2, \cdots, K\}$).

II. PRELIMINARIES

We briefly overview the key techniques, such as reinforcement learning (RL) and the g_m/I_D methodology.

A. REINFORCEMENT LEARNING (RL)

RL is a subset of machine learning technologies that excels in optimizing and/or controlling with large search spaces [20], [23]. RL consists of two main components: the environment and the agent. Firstly, the environment contains information reflecting the nature of a given field, serving as a repository for models such as simulators. The environment produces a satisfactory reward to attain the desired objective as well. Secondly, the agent optimizes its sequential behaviors through iterative experiences with the environment. The iterative process is referred to as trajectory. Each trajectory returns a cumulative reward that is exploited to optimize the agent's behavior. Through these continuous interactions, the agent gains a comprehensive understanding of the environment.

Theoretically, RL can be conceptualized as a Markov decision process (MDP), which is a decision-making process adhering to the Markov property. An MDP can be represented as a tuple of $\langle \mathcal{S}, \mathcal{A}, \mathbf{P}, r, \gamma \rangle$, where each component corresponds to: the state space, $\mathcal{S}$; the action space, $\mathcal{A}$; the transition kernel, $\mathbf{P} = (p^a_{s,s'})$; the reward function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$; and the discount factor $\gamma \in [0, 1]$.

At each time step t, an agent observes a state $s_t \in \mathcal{S}$ from the environment and executes an action $a_t \in \mathcal{A}$ guided by its trained policy $\pi : \mathcal{S} \to \mathcal{A}$. Subsequently, according to the transition kernel $p(s_{t+1} \mid s_t, a_t) = p^a_{s,s'}$, the environment evolves to the next time step and returns the next state $s_{t+1} \in \mathcal{S}$. Given a policy π, the value and the action-value functions are respectively evaluated as

$$V_\pi(s_t) = \mathbb{E}_{a_t, s_{t+1}, \cdots}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l})\right] \tag{1}$$

$$Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, \cdots}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, r_{t+l})\right], \tag{2}$$

where $a_t \sim \pi(\cdot \mid s_t)$ and $s_{t+1} \sim p^{a_t}_{s_t, s_{t+1}}$. Given an MDP, the objective of RL is to find an optimal policy $\pi^\star$ such that

$$\pi^\star = \arg\max_\pi V_\pi(s), \ \forall s \in \mathcal{S}. \tag{3}$$

In (model-free) RL, which is considered in this paper, the agent's goal is to solve the aforementioned optimization problem only using samples (or episodes) without the knowledge of an environment [20]. Deep RL (DRL) refers to approximating the value or policy function via a deep neural network (DNN), which is required to handle continuous state/action spaces.

B. THE G_M/I_D METHODOLOGY

Historically, in equation-based circuit automation methods, circuit designers have employed device equations to describe the physical aspects of primitive devices such as a Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET) for their operating points. For instance, the drain current of a long-channel N-type MOSFET (NMOS) can be computed using the following equation [24]:

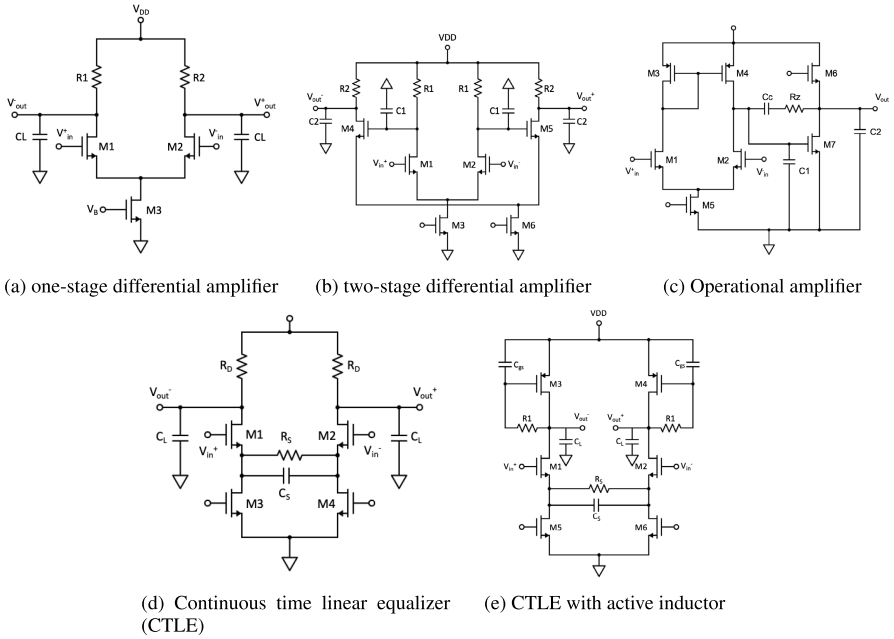
$$I_D = \frac{1}{2}\mu_n C_{\text{ox}}\left(\frac{W}{L}\right)(V_{\text{GS}} - V_{\text{th}})^2. \tag{4}$$

The equation incorporates various technological parameters, such as mobility (μ_n), oxide thickness (C_{ox}), threshold voltage (V_{th}), and the bias conditions (V_{GS}, I_D). Then the small-signal transconductance (g_m) and the sizing parameters (W/L) can be expressed by the following derivative formulas:

$$g_m = \frac{\partial I_D}{\partial V_{\text{GS}}} = \frac{2I_D}{V_{\text{GS}} - V_{\text{th}}}, \tag{5}$$

and

$$\frac{W}{L} = \frac{2I_D}{\mu_n C_{\text{ox}}(V_{\text{GS}} - V_{\text{th}})^2}. \tag{6}$$



(a) one-stage differential amplifier (b) two-stage differential amplifier (c) Operational amplifier

(d) Continuous time linear equalizer (CTLE) (e) CTLE with active inductor

FIGURE 1. Example topologies that can easily be formulated into the MOCO problem. (a) one-stage differential amplifier, (b) two-stage fully differential amplifier, (c) operational transconductance amplifier, (d) continuous time linear equalizer (CTLE), and (e) CTLE circuit with active inductor [22].

For a desired bias point $(V_{GS} - V_{th})$, the drain current I_D can be determined by combining the equation (5) with the desired g_m, which is set according to its specific requirement such as the circuit's Gain-Bandwidth Product (GBW). Subsequently, the sizing parameters can be obtained by the equation (6) based on the computed I_D.

One limitation associated with the traditional approach is that the expression (4), along with its derivative formulas (5) and (6), do not hold in the context of modern CMOS technology. Therefore, a more applicable approach has been proposed: the g_m/I_D method [2], [25], [26]. The main idea of the g_m/I_D method can be explained as follows: since both g_m and I_D are proportional to W/L, the ratio of the two elements is independent of the sizing parameters [2], [25], [26]. This ratio depends solely on the device's DC bias conditions, which can be defined as a transconductance efficiency [2], [26]. For example, the transconductance efficiency of a long-channel MOSFET is expressed as follows:

$$\frac{g_m}{I_D} = \sqrt{\frac{2}{I_D}\left(\frac{\mu}{C_{ox}}\right)\left(\frac{W}{L}\right)} = \frac{2}{V_{GS} - V_{th}}. \tag{7}$$

It should be noted that the second and third terms in the equation (7) are not applicable in modern CMOS technologies. Instead, the numerical value of g_m/I_D for the device's bias point is derived from simulation. Various studies have shown the advantage of the g_m/I_D methodology in terms of sizing accuracy compared to traditional approaches [2], [26], [27]. The methodology generally offers a unified sizing procedure for MOS devices for a wide spectrum of inversion regions, including the subthreshold region for low-power circuits and the strong-inversion region for high-bandwidth circuits.

III. PROBLEM FORMULATION

We aim to design an analog circuit to achieve a target specification with domain observations that experts can efficiently extract or observe. This problem can be formulated as a multi-objective combinatorial optimization (MOCO). Given the circuit topology and the so-called characteristic information $\mathbf{c} \in \mathcal{C}$, we aim to optimize the sizing parameter $\mathbf{q} \in \mathbb{Z}_K^N$ that optimizes the given specifications.

Fig. 1 shows five different topologies considered in this paper that can be formulated into the derived MOCO problem. In this study, for example, we focused on optimizing the current consumption for the target GBW to verify its capability and find an optimal solution in 1. Therefore, the objective functions are determined by the circuit topology and the characteristic vector $\mathbf{c} \in \mathcal{C}$:

$$f_{\epsilon,1}(\mathbf{q}; \mathbf{c}) \triangleq f_1(\mathbf{q} + \boldsymbol{\epsilon}_1; \mathbf{c}) = -I_D \tag{8}$$

$$f_{\epsilon,2}(\mathbf{q}; \mathbf{c}) \triangleq f_2(\mathbf{q} + \boldsymbol{\epsilon}_2; \mathbf{c}) = \text{GBW}. \tag{9}$$

However, depending on their applications, amplifiers have additional parameters, such as noise and power supply rejection ratio (PSRR). As shown in (8) and (9), given the circuit topology, the objective functions f_i's can be varied according to the choices of $\mathbf{c} \in \mathcal{C}$. Specifically, each $f_i : \mathcal{Q} \to \mathbb{R}$ is determined according to the circuit topology and $\mathbf{c} \in \mathcal{C}$ is a fixed *characteristic* vector (or possibly scalar) that subtly intimates the input-output relationship of the given topology. In the above optimization, $\mathbf{c}$ is fixed, provided by analog-circuit experts before tackling the

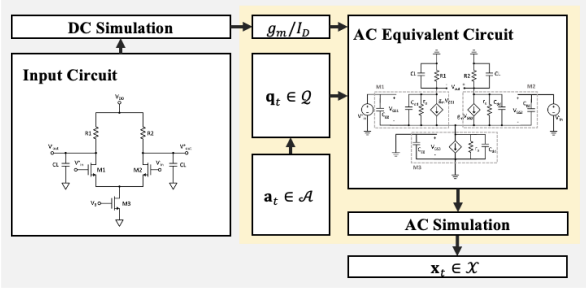


FIGURE 2. The function operator between parameter and observation. The agents possess knowledge solely based on their observations, referred to as x. They lack any understanding of the physical characteristics of the structure.

above MOCO. $\mathcal{C}$ is also associated with a given topology. Given the topology in the example above, $\mathbf{c} \in \mathcal{C}$ can capture observations such as load capacitance ratio ($c = C_1/C_2$). Note that each component of $\mathbf{q}$ is discrete, even if the corresponding parameter of the analog circuit is continuous, such as width, capacitances, etc. Without the quantizations, the optimization problem is not manageable at all as the objective functions f_i's are too complex and are not even mathematically well-defined. Also, $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$ can capture the uncertainties arising from various non-ideal factors in the analog circuit, including the complex I-V characteristics that make the MOSFET operate in various regions elsewhere its linear region, exhibited by modern CMOS transistors and layout-dependent effects such as parasitic capacitance. Then, the optimization problem to design the analog circuit can be formulated as

$$\begin{aligned} \max \quad & \big(f_{\epsilon,1}(\mathbf{q};\mathbf{c}), f_{\epsilon,2}(\mathbf{q};\mathbf{c})\big) \\ \text{subject to} \quad & \mathbf{q} \in \mathbb{Z}_K^N. \end{aligned} \tag{10}$$

This is the complex MOCO problem under *uncertainty*, which is intractable due to the growth in complexity of topologies, the complicated structures of f_m's, and the difficulty of handling uncertainties.

Fig. 2 shows the relationship between the parameter $\mathbf{q}_t$ and the observation $\mathbf{x}_t$ of the overall flow of the framework. First, we overcome the challenging uncertainty problem via the g_m/I_D methodology, which enables capturing the uncertain factors through precomputed parameters such as the transconductance efficiency (g_m/I_D) and normalized parasitic capacitances (C_{dd}/I_d and C_{gg}/I_D). By conducting DC simulations, these parameters can be obtained easily, and consequently, the parameter optimization of the analog circuit can be performed only with the AC model. Thus, the uncertainties in (8) and (9) can be eliminated. Leveraging the g_m/I_D methodology, thus, the optimization problem in (10) can be simplified as

$$\begin{aligned} \max \quad & \big(f_1(\mathbf{q};\mathbf{c}), f_2(\mathbf{q};\mathbf{c})\big) \\ \text{subject to} \quad & \mathbf{q} \in \mathbb{Z}_K^N, . \end{aligned} \tag{11}$$

In general, the design problem of an analog circuit based on g_m/I_D methodology can be formulated as MOCO:

$$\begin{aligned} \max \quad & \big(f_1(\mathbf{q};\mathbf{c}), f_2(\mathbf{q};\mathbf{c}), \ldots, f_M(\mathbf{q};\mathbf{c})\big) \\ \text{subject to} \quad & \mathbf{q} \in \mathbb{Z}_K^N. \end{aligned} \tag{12}$$

The most widely used way to handle MOCO is the preference-based scalarization [28]. For an M-objective MOCO, a preference vector for the objective functions can be defined as $\mathbf{w} \in \mathbb{R}^M$ that satisfies $w_m \geq 0$ and $\sum_{m=1}^M w_m = 1$. Given a preference vector $\mathbf{w}$, the optimization problem in (12) can be reformulated as

$$\begin{aligned} \max \quad & \mathbf{w}^\mathsf{T}\mathbf{f}(\mathbf{q};\mathbf{c}) \\ \text{subject to} \quad & \mathbf{q} \in \mathbb{Z}_K^N \end{aligned} \tag{13}$$

where $\mathbf{f}(\mathbf{q};\mathbf{c}) = [f_1(\mathbf{q};\mathbf{c}), \cdots, f_M(\mathbf{q};\mathbf{c})]^\mathsf{T}$. Here, the function $\mathbf{f}(\mathbf{q};\mathbf{c})$ cannot always be known, and the search space of $\mathbf{q}$ is extremely large, making the problem NP-hard. Therefore, it is required to develop an efficient algorithm to solve the CO problem in (13) without requiring the exact function $\mathbf{f}$. We propose a sequential framework that exploits a pre-trained RL agent to handle this continuing problem that does not include a terminal point. Each iteration includes an episodic task with a terminal point, referred to as the *design specifications*. In the next section, we explain the details of our sequential framework and how the formulated MOCO problem can be solved via a sequential RL framework, and we introduce how the agents were trained with various types of specifications.

IV. PROPOSED METHOD

This section proposes an efficient RL-based method to solve the CO problem in (13) with a given preference vector $\mathbf{w}$. To handle the CO problem favorably with an RL agent, we decompose it into subproblems (or feasible problems). This decomposition is largely motivated by the fact that the RL framework can tackle episodic tasks (with terminal states) more efficiently than continuing tasks [20]. In the context of the RL, the maximization in (13) can be formulated as continuing tasks, whereas the feasible problems can be defined as episodic tasks. In each feasible problem, all feasible solutions are assigned to the terminal states. In the subsequent subsections, we first provide a detailed description of the proposed sequential (or decomposition) approach to exploit an RL agent's searching ability. Then, we explain how to train such RL agents by setting up an environment and defining a specific RL algorithm.

A. SEQUENTIAL FRAMEWORK

In the proposed sequential framework, the complex CO problem in (13) is decomposed into a sequence of subproblems. Here, the i-th subproblem is simply formulated as the feasible problem and finds a feasible solution belonging to a feasible region $\mathcal{F}_i$. To define a feasible region that sequentially matches the objective of the CO problem, we introduce a design specification vector $\mathbf{x}_i^\star \in \mathcal{X}^\star \subseteq \mathbb{Z}_K^N$, where

$\mathcal{X}^\star$ contains all possible design specifications of a target circuit topology. For each subproblem i, $\mathbf{x}_i^\star$ is selected such that

$$\mathbf{w}^\mathsf{T}\mathbf{x}_i^\star \geq \mathbf{w}^\mathsf{T}\mathbf{x}_{i-1}^\star, \tag{14}$$

where the specification vector $\mathbf{x}_{i-1}^\star$ is selected in the previous subproblem, which serves as the constraint of the RL agent. We expect that the sequence $\mathbf{w}^\mathsf{T}\mathbf{x}_1^\star, \mathbf{w}^\mathsf{T}\mathbf{x}_2^\star, \ldots$ can approach the upper-bound (i.e., the optimal solution of our problem in (13)): as i grows, $\mathbf{w}^\mathsf{T}\mathbf{x}_i^\star$ can converge to $\max \mathbf{w}^\mathsf{T}\mathbf{f}(\mathbf{q};\mathbf{c})$. Accordingly, the solution of the feasible problem $\mathcal{F}_i$ can approach the optimal parameter $\mathbf{q}_\star = \arg\max \mathbf{w}^\mathsf{T}\mathbf{f}(\mathbf{q};\mathbf{c})$. Based on this, we define a feasible region $\mathcal{F}_i$ for the i-th subproblem:

$$\mathcal{F}_i \triangleq \left\{\mathbf{q} \in \mathbb{Z}_K^N : \mathbf{w}^\mathsf{T}\mathbf{d}(\mathbf{q}, \mathbf{x}_i^\star) \leq \eta\right\}, \tag{15}$$

where $\eta > 0$ is a hyperparameter to determine the size of the feasible set,

$$\mathbf{d}(\mathbf{q}, \mathbf{x}_i^\star) = \left[d_1(f_1(\mathbf{q};\mathbf{c}), x_{i,1}^\star), \cdots, d_M(f_M(\mathbf{q};\mathbf{c}), x_{i,M}^\star)\right]^\mathsf{T}, \tag{16}$$

and $d_m(f_m(\mathbf{q};\mathbf{c}), x_{i,m}^\star)$ denotes a distance measure suitable for the m-th objective function f_m.

Example 4.1: Consider the amplifier with a feedback loop in Fig. 1c. In this example, the specification vector is defined as

$$\mathbf{x}_i^\star = [I_i^\star, \mathrm{GBW}_i^\star, \mathrm{PM}_i^\star]. \tag{17}$$

In this case, we use the following distance metrics:

$$d_1(f_1(\mathbf{q};\mathbf{c}), I_i^\star) = \max\left\{\frac{f_1(\mathbf{q};\mathbf{c}) - I_i^\star}{f_1(\mathbf{q};\mathbf{c}) + I_i^\star}, 0\right\} \tag{18}$$

$$d_2(f_2(\mathbf{q};\mathbf{c}), \mathrm{GBW}_i^\star) = \max\left\{\frac{\mathrm{GBW}_i^\star - f_2(\mathbf{q};\mathbf{c})}{\mathrm{GBW}_i^\star + f_2(\mathbf{q};\mathbf{c})}, 0\right\} \tag{19}$$

$$d_3(f_3(\mathbf{q};\mathbf{c}), \mathrm{PM}_i^\star) = \left|\frac{f_3(\mathbf{q};\mathbf{c}) - \mathrm{PM}_i^\star}{f_3(\mathbf{q};\mathbf{c}) + \mathrm{PM}_i^\star}\right|. \tag{20}$$

Our RL agent is trained to solve such feasible problems efficiently. This agent can generate a sequence of states (i.e., the parameters) by taking actions from a learned policy (i.e., π_θ). When one of the terminal states (i.e., one of the feasible points) is reached, the RL agent stops to take action. Then, we can obtain the feasible solution (say, $\mathbf{q}_i \in \mathcal{F}_i$) from the final state. The outline of the proposed method is as follows:

- Given the feasible region $\mathcal{F}_i$ (i.e., the specification vector $\mathbf{x}_i^\star$), the RL agent seeks a feasible solution by taking a sequence of actions from a learned policy.
- Once a feasible solution is found (i.e., $\mathbf{q}_i \in \mathcal{F}_i$), the next subproblem is defined with a *refined* feasible set $\mathcal{F}_{i+1} \subseteq \mathcal{F}_i$ by evolving $\mathbf{x}_i^\star$ into $\mathbf{x}_{i+1}^\star$.
- The above process is repeated until a near-optimal solution is attained.

By constructions of the feasible sets, we expect that $\mathbf{w}^\mathsf{T}\mathbf{x}_1^\star \leq \mathbf{w}^\mathsf{T}\mathbf{x}_2^\star \leq \cdots \leq \mathbf{w}^\mathsf{T}\mathbf{x}_{I_{\max}}^\star \leq \max \mathbf{w}^\mathsf{T}\mathbf{f}(\mathbf{q};\mathbf{c})$, i.e., $\mathbf{w}^\mathsf{T}\mathbf{x}_i^\star$ approaches the maximum value (i.e., $\max \mathbf{w}^\mathsf{T}\mathbf{f}(\mathbf{q};\mathbf{c})$) as i grows. Thus, the RL agent can yield a good solution to the CO problem in (13). Specifically, the sequence of feasible regions is constructed as follows. For each iteration i, the pre-trained RL agent (or policy) iterates through a fixed length of the time step T and explores the search space. Once the RL agent finds a parameter belonging to the i-th feasibility region $\mathcal{F}_i$, our algorithm moves to the next subproblem by constructing a *refined* feasible reason. For this, the design specification vector $\mathbf{x}_{i+1}^\star$ is determined as

$$\mathbf{x}_{i+1}^\star = \mathbf{x}_i^\star + \boldsymbol{\delta}_i^+, \tag{21}$$

for some positive vector $\boldsymbol{\delta}_i > 0$ and accordingly, the subproblem $i+1$ with a harder task is defined with the following feasible region:

$$\mathcal{F}_{i+1} = \left\{\mathbf{q} \in \mathbb{Z}_K^N : \mathbf{w}^\mathsf{T}\mathbf{d}(\mathbf{q}, \mathbf{x}_{i+1}^\star) \leq \eta\right\}. \tag{22}$$

Otherwise, the subproblem $i+1$ with an easier task is defined with the design specification vector:

$$\mathbf{x}_{i+1}^\star = \mathbf{x}_i^\star - \boldsymbol{\delta}_i^+. \tag{23}$$

Note that the target rate $\boldsymbol{\delta}_i^+ > 0$ is a hyperparameter and is carefully chosen based on the domain knowledge of analog-circuit experts. The detailed procedures of the proposed fRL-AD are provided in Algorithm 1.

B. TRAINING AN RL AGENT

We describe how to train our RL agent to tackle feasible problems efficiently. Note that feasible problems can be changed according to the circuit topology and the characteristic vector $\mathbf{c}$. We remark that our learned agent can solve the feasible problems for various $\mathbf{c}$ under the fixed topology. Namely, we only need to train a new RL agent for each new topology, irrespective of characteristic vectors.

Regarding the RL framework, we adopt a policy optimization strategy that optimizes the parameters under the g_m/I_D methodology. We employ an online RL algorithm to train our RL agent and the proximal policy optimization (PPO) algorithm. Notably, the PPO algorithm demonstrates its strength in convergence and stability, courtesy of using importance sampling and the clipping function [29] as a practical way of sufficing the Kullback-Leibler (KL) divergence constraint. We emphasize that the proposed sequential framework in Section IV-A can be integrated with other RL algorithms such as deep Q-network (DQN), advantage actor-critic (A2C) that does not require constraints for RL algorithms., or soft actor-critic (SAC) [30], [31], [32], [33] that adopts entropy constraints to enhance exploration property.

Given that the proposed method adheres to the RL framework, it is essential to meticulously design an environment consisting of the three main components: state, action, and reward. Details of the three components are illustrated in the following. For ease of explanation, we omit the subscript i and only consider the subscript t.

Algorithm 1 fRL-AD

Input: Pre-trained RL agent (or learned polcy) π_θ, maximum iteration $I_{\max}$, maximum time-step T, fixed topology (with g_m/I_D methodology) f, preference vector $\mathbf{w}$, distance measures $\{d_m : m \in [M]\}$, and target rates $\{\boldsymbol{\delta}_i : i \in [I_{\max}]\}$.

Initialization: $\mathbf{q}_{0,\text{init}} = \mathbf{1}$, sample $\mathbf{x}_0^\star \sim \mathcal{X}^\star$, $\mathbf{s}_{0,\text{init}}$ using (25), and feasible region $\mathcal{F}_0 = \{\mathbf{q} \in \mathbb{Z}_K^N : \mathbf{w}^\mathsf{T}\mathbf{d}(\mathbf{q}, \mathbf{x}_0^\star)\}$.

for $i = 1, 2, \cdots, I_{\max}$ **do**
  for $t = 1, 2, \cdots, T$ **do**
    sample action $\mathbf{a}_{t-1} \sim \pi_\theta(\cdot \mid \mathbf{s}_{i-1,t-1})$
    update $\mathbf{q}_{i-1,t} \leftarrow \mathbf{q}_{i-1,t-1} + \mathbf{a}_{t-1}$
    if $\mathbf{q}_{i-1,t} \in \mathcal{F}_{i-1}$ via (15) **then**
      update $\mathbf{q}_{i,0} \leftarrow \mathbf{q}_{i-1,t}$
      update $\mathbf{x}_i^\star = \mathbf{x}_{i-1}^\star + \boldsymbol{\delta}_i^+$ via (21)
      end iteration $i-1$
    end if
  end for
  if $t = T$ **then**
    update $\mathbf{x}_i^\star = \mathbf{x}_{i-1}^\star - \boldsymbol{\delta}_i^+$ via (23)
  end if
end for

Output: $\mathbf{q}_{I_{\max}}, \mathbf{x}_{I_{\max}}$

1) STATE

Let $\mathcal{S}$ be the state space consisting of state vectors $\mathbf{s} \in \mathcal{S}$. In the proposed RL framework, it is defined as

$$\mathcal{S} = \mathcal{X} \times \mathcal{X}^\star \times \mathcal{Q} \times \mathcal{C} \tag{24}$$

where $\times$ denotes the cartesian product of sets. Here, $\mathcal{Q} = \mathbb{Z}_K^N$ is a set of vectors that contain N sizing parameters for a given topology, $\mathcal{X} = \mathbb{R}^M$ is a set of the observation vectors containing the M output results obtained from the circuit simulator when the parameter $\mathbf{q}_t$ is given, $\mathcal{X}^\star \subset \mathbb{R}^M$ is the set of vectors containing each trajectory's design specifications that directly determines the feasibility set $\mathcal{F}$, $\mathcal{C} = \mathbb{R}^D$ is the set of the characteristic vector that cover diversity of a given topology. In this paper, $\mathcal{C}$ is the set of (shifted and normalized) ratios between loaded capacitance between stages in multi-stage amplifiers, and D should be $\binom{n}{2}$ for such topologies where n is the number of stages. Then, the state vector at time step t (i.e., $\mathbf{s}_t$), which represents the environment's status at time step t and serves as an input for the policy network, can be defined as

$$\mathbf{s}_t = \left[\mathbf{x}_t^\mathsf{T}, \mathbf{x}^{\star\mathsf{T}}, \mathbf{q}_t^\mathsf{T}, \mathbf{c}^\mathsf{T}\right]^\mathsf{T} \in \mathcal{S}. \tag{25}$$

Let $\mathcal{S}_T \subset \mathcal{S}$ be the terminal state space, i.e.,

$$\mathcal{S}_T = \{\mathbf{s}_t \in \mathcal{S} \mid \mathbf{q}_t \in \mathcal{F}\}, \tag{26}$$

where the operator $\geq$ between vectors denotes the elementwise inequality between two vectors. It contains the state vectors whose corresponding topology outputs belong to the feasible region $\mathcal{F}$. Given an action $\mathbf{a}_t$ and the current state $\mathbf{s}_t$, the state transition is performed as

$$\mathbf{s}_{t+1} = \left[\mathbf{x}_{t+1}^\mathsf{T}, \mathbf{x}^{\star\mathsf{T}}, \mathbf{q}_{t+1}^\mathsf{T}, \mathbf{c}^\mathsf{T}\right]^\mathsf{T}, \tag{27}$$

where $\mathbf{q}_{t+1} = \mathbf{q}_t + \mathbf{a}_t$ and $\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{q}_{t+1}; \mathbf{c})$. Since $f(\cdot; \mathbf{c})$ is unknown, $\mathbf{x}_{t+1}$ is directly obtained from the environment (or simulator).

2) ACTION

Based on the value of the state vector ($\mathbf{s}_t$), the agent samples an action from a trained policy network (i.e., $\mathbf{a}_t \sim \pi(\cdot|\mathbf{s}_t)$). In the proposed framework, the action corresponds to the adjustment of sizing parameters. To mimic the decision-making process of human designers when adjusting the sizing parameters, we initially defined the action space to consist of three choices for each parameter: increase, decrease, or maintain the current value of the sizing parameter ($\mathbf{q}_t$) [17]. In this paper, we divide the action space into two categories according to their dependency on the output performance: *fixed action space* and *adaptive action space*.

a: FIXED ACTION SPACE

For an environment defined with the *fixed action space*, the agent adjusts the sizing parameter vector with a pre-defined magnitude that remains constant (usually 1) throughout the episodes [14], [17]. The fixed action space can be defined as a multi-dimensional discrete action space, and the dimension of the action space is identical to that of the sizing parameter space $\mathcal{Q}$:

$$\mathcal{A} = \{-1, 0, 1\}^M. \tag{28}$$

Each element corresponds to a specific action for the fixed action space: -1 indicates decreasing the parameter value, 1 indicates increasing the parameter, and 0 indicates maintaining the current value. However, this definition of action space cannot consider the proximity between the circuit output ($\mathbf{x}_t \in \mathcal{X}$) and the design specifications ($\mathbf{x}^\star \in \mathcal{X}^\star$). we propose an action framework that can adjust the step size concerning the difference between $\mathbf{x}_t$ and $\mathbf{x}^\star$, denoted as the *adaptive action*, which is explained in the following subsection.

b: ADAPTIVE ACTION SPACE

To address the slow optimization speed, we draw inspiration from human designers who take bold actions when there is a significant difference between the desired specifications and the current state. Specifically, our approach actively adjusts the action size to encourage the agent to take bolder actions in such cases and reach the specifications faster. This is achieved through the proposed *adaptive action space* where we denote each action space using a subscript t since the action space differs every time step (i.e., $\mathcal{A}_t$):

$$\mathcal{A}_t = \{-\Delta_t, 0, \Delta_t\}^M, \quad \text{where,} \quad \Delta_t = \max(1, \lfloor \alpha r_t \rfloor) \tag{29}$$

where $\lfloor \cdot \rfloor$ denote the floor function since $\mathbf{q}$ is an integer.

By adjusting the parameter α, the designer can control the trade-off between convergence speed and stability. Indeed, the *adaptive action space* can be a general case of the fixed counterpart because as α grows near zero, the expression for every time step t converges to the fixed action space $\mathcal{A}$ regardless of the performance. We numerically show that our proposed *adaptive action space* enhances step efficiency.

3) REWARD

The reward function $r(\mathbf{s}_t, \mathbf{a}_t)$ significantly affects the convergence rate and accuracy of the agent. Policy optimization methods aim to maximize the cumulative reward in each episode (i.e., the episode reward mean of all workers). The proposed analog circuit designing environment has design specifications, and to address the design specifications with various performance measures, we generalize the reward function from [17] using the designed metric functions:

$$r(\mathbf{s}_t, \mathbf{a}_t) = \begin{cases} \sum_{m=1}^{M} w_m d_m(f_m(\mathbf{q}_{t+1}; \mathbf{c}), x_m^\star), & \mathbf{s}_{t+1} \in S \backslash S_T \\ 10, & \mathbf{s}_{t+1} \in S_T \end{cases} \tag{30}$$

Here, w_m is a hyperparameter that indicates the priority of each output performance. For outputs throughout this paper, we mainly deal with three different types of design constraints: upper-bound constraints, lower-bound constraints, and error constraints. Upper-bound constraints are design specifications where the output should be under the given specification, and design specifications such as power consumption could be an example. Similarly, lower-bound constraints design specifications that should be lower than the output; for example, a GBW could be a lower-bound constraint. Finally, error constraints indicate design specifications where the output should be near or equal to the specification, and the phase margin of an operational amplifier generally can be an example since the phase margin generally aims to be near 60°.

For example, the reward function $r(\mathbf{s}_t, \mathbf{a}_t)$ of the lower bound constraint is depicted in Fig. 3. The step reward would increase as the state approaches the desired lower bound constraint and eventually returns a positive number 10 as the observed value surpasses the given lower bound constraint with some threshold η. This shows that the reward function returns a value closely related to suffice the given sequential constraint problems. This reward function generates a cumulative reward for designing a circuit that meets certain specifications.

Fig. 4 demonstrates the total block diagram of the proposed fRL-AD framework. Using the three major components introduced above, the agent is trained using the reward function. After the training phase, the pre-trained agent can select a plausible action according to the neural network and its observations. In the subsequent subsections, we provide an elaborate explanation of both the training and validation phases of the proposed fRL-AD framework.

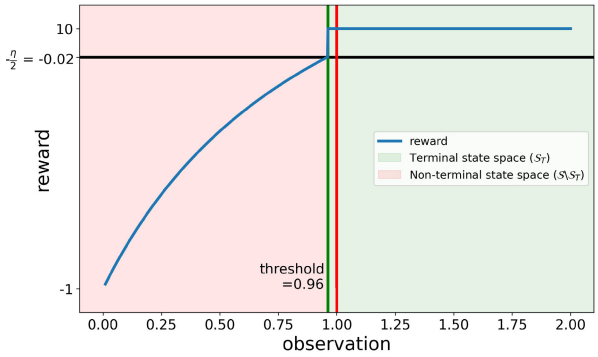
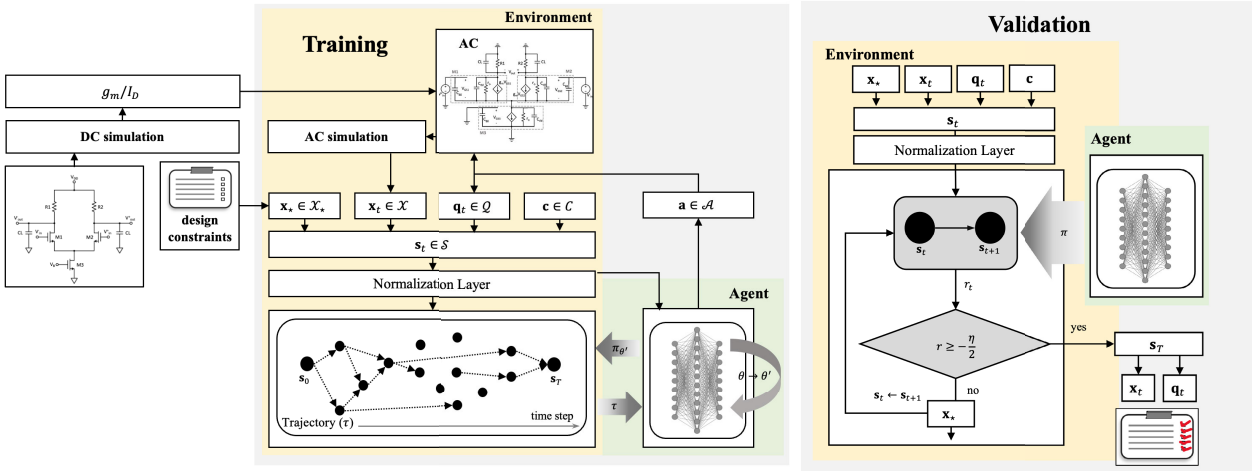


FIGURE 3. Reward function and terminal state space (S_T). S_T is highly related to the reward function $r(\mathbf{s}_t, \mathbf{a}_t)$.

C. TRAINING

In the training phase, design specifications should be given within a practical boundary to train an RL agent to conduct complicated analog circuit designing tasks without retraining. The practical boundaries are given by expert knowledge, which can benefit the agent in learning multiple specifications within the boundary. The process of providing the agent with diverse design specifications and generating trajectories with each constraint is illustrated in Fig. 4a. These specifications can be categorized into three types based on their difficulty: *hard specifications*, *fit specifications*, and *loose specifications*. *Hard specifications* represent design requirements that are hard or impossible to achieve within the given circuit topology, resulting in output performances that can not drop inside the feasibility region $\mathcal{F}$ (or equivalently, the terminal state space $\mathcal{S}_T$). *Fit specifications*, on the other hand, are challenging design requirements that can only be met if the agent's policy is nearly optimal. Achieving *fit specifications* requires the agent to navigate the solution space effectively. *Loose specifications* refer to requirements that the agent can achieve with less effort compared to *fit specifications*. During training, the agent is exposed to all three types of specifications. Furthermore, The initial states ($\mathbf{s}_0$) are uniformly selected from the state space S. Selecting initial states uniformly from the total space guarantees fast exploration of the environment and enhances the agent's probability of convergence to the optimal policy.

Finally, to consider a wide range of device parasitic, we incorporate the small-signal parameter C_{dd}/I_D into the simulator while remaining the agent uninformed about the incorporated value of the parasitic capacitance. For example, in differential amplifiers, this simulation setting seems to enable the agent to consider parasitic capacitance by considering the relationship between the design parameters and the designed output performance and whether to decide on an action to consume more power to drive the given load capacitance. Section V shows the agent's capability to consider parasitic capacitance by providing simulation results.



(a) Training block diagram of fRL-AD

(b) Validation block diagram of fRL-AD

FIGURE 4. Block diagram of fRL-AD. (a) The training phase shows how the framework trains a policy. (b) The validation phase shows how the trained policy designs a desired specification.

D. VALIDATION

In the validation phase, as depicted in Fig. 4b, we randomly select a specification within the given range of design specifications. The specification range is selected wide enough to cover various difficulty levels. Fig. 4b also shows that with the given specification ($\mathbf{x}^\star \in \mathcal{X}^\star$) and the circuit observation at the initial time ($\mathbf{x}_{\text{init}} \in \mathcal{X}$), the initial state $\mathbf{s}_0$ can be defined as

$$\mathbf{s}_0 = [\mathbf{x}_{\text{init}}, \mathbf{x}^\star, \mathbf{q}_{\text{init}}, \mathbf{c}]. \tag{31}$$

Then with the given state ($\mathbf{s}_t$), the agent samples an action ($\mathbf{a}_t$) from the learned policy $\pi_\theta(\cdot \mid \mathbf{s}_t)$ and computes the corresponding reward $r(\mathbf{s}_t, \mathbf{a}_t)$. The well-designed reward function, which inherently incorporates the percentage accuracy, allows the environment to determine whether the state is in the terminal. If the state is not in the terminal, the sizing parameter $\mathbf{q}_t \in \mathcal{Q}$ changes to $\mathbf{q}_{t+1} \in \mathcal{Q}$ based on the sampled action $\mathbf{a}_t$, which in turn updates the current circuit observation $\mathbf{x}_{t+1}$. This process constructs the next state $\mathbf{s}_{t+1}$ and iterates until the parameter combinations drop into the feasible region $\mathcal{F}$. Once the terminal state is reached, we extract the first to M-th elements of the state vector that represent the current performance and the $(2M+1)$-th to $(2M+N)$-th elements of the state for the sizing parameters of the circuit designed by the trained RL agent. These elements provide a summary of the circuit's achieved performance. Since the formulated MDP is discrete and finite, an optimal deterministic policy would exist given that the policy is trained sufficiently [20], [34]. This proof eventually makes the problem equivalent to finding the shortest path to the best possible combination. Therefore, the convergence time of each validation depends linearly on the size of each parameter space and the number of parameters. For example, for a topology with N sizing parameters, each quantized by K level would have the maximum convergence time $\mathcal{O}(KN)$.

Furthermore, this validation method continues until a certain number of iterations (I) is met by updating the initial parameters ($\mathbf{q}_{\text{init}}$) and the feasible region ($\mathcal{F}$) at every iteration. Therefore, the total convergence time would be $\mathcal{O}(IKN)$. Note that this convergence time ensures the convergence to the parameter combination that returns the highest reward, where the reward (30) is assumed to be eligible according to the distance measure (16). In the following sections, we have numerically shown that our trained agents' validation time falls into the convergence time above, where the validation reward nearly converges.

V. EXPERIMENT

As shown in Fig. 1, Several topologies were used to train and validate the proposed framework: a single-stage differential amplifier, a two-stage amplifier, an operational transconductance amplifier, and a continuous time linear equalizer (CTLE). During training, we used the PPO algorithm to update the policy network. To efficiently update the policy network, we randomly selected initial conditions uniformly distributed along the sizing parameter space. This random initialization ensures exploration, which is essential for the agent to experience various states during training.

We present the simulation results derived from our proposed framework. These results are segmented into two primary categories. Firstly, we exhibit the resilience of our proposed method through simulations performed under device parasitic uncertainty. The agents are trained and validated through rigorous testing, demonstrating our method's consideration of normalized parasitic components. As a result, the agent can design circuits capable of handling parasitic uncertainty, eliminating the need for supplementary frameworks.

Secondly, we introduce experimental results for environments employing the *adaptive action space* during the training and validation phases. To contrast the outcomes of adaptive and fixed action spaces, we executed these tests

in a relatively simpler setting than the first experiment. We conducted 100 identical episodes to compare the average performance of agents across both environments. It is important to note that even when settings were identical to the first experiment, the same advantages were observed when the *adaptive action* strategy was employed. This suggests that the efficacy of our *adaptive action* technique is independent of the simulation settings.

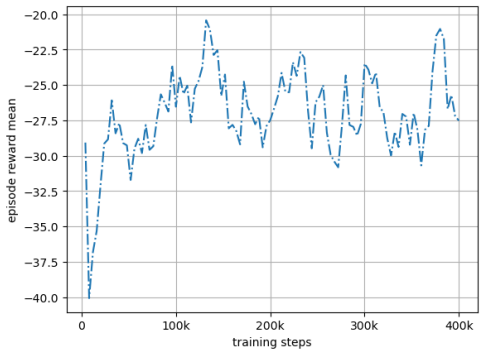
A. DIFFERENTIAL AMPLIFIER

As shown in Fig. 1a, we first experiment with single-stage differential amplifiers using the SPICE simulator. This topology is introduced to measure the agent's basic capabilities to optimize the sizing parameters for the target performance parameter with minimal power consumption. The current observation of the circuit $\mathbf{x}_t$ is a vector containing the total current values and GBW at time step t ($I_t \in [0.01, 2]$ mA and $\text{GBW}_t \in [0.01, 10]$ Grad/s). The design goal $\mathbf{x}^\star = [I^\star, \text{GBW}^\star]$ in the differential amplifier is sampled from the same interval as I_t, where the subscript $\star$ denotes the desired upper- or lower-bound constraints. We can put the observation $\mathbf{x}_t$ and sizing parameter $\mathbf{q}_t$ together since it is identical thanks to the g_m/I_d methodology. Furthermore, for a single-stage differential amplifier, the dimension of $\mathbf{c}$ (D) is zero since $\binom{1}{2} = 0$ and can be omitted from the state vector.Therefore, the state $\mathbf{s}_t \in S$ is defined with

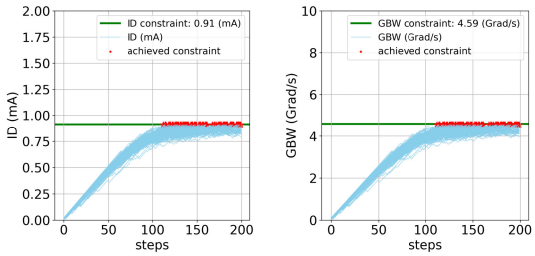
$$\begin{aligned} \mathbf{x}_t &= [I_t, \text{GBW}]^\mathsf{T} \\ \mathbf{x}^\star &= \left[I^\star, \text{GBW}^\star\right]^\mathsf{T} \\ \mathbf{q}_t &= [I_t] \\ c_r &= 0. \end{aligned} \tag{32}$$

Since we apply the g_m/I_D method, which considers the bias currents as the main design variables, the action consists of directly adjusting the bias currents of the amplifier. Therefore, the action is a one-dimensional scalar from a one-dimensional discrete action space: $a \in \mathcal{A} = \{-1, 0, 1\}$. We initialize K as 200 to quantize the sizing parameter space into K linear spaces, and therefore, each action corresponds to adjusting the current I by 0.01 mA for the *fixed action space*. It is important to note that the simple differential amplifier applying the g_m/I_D method is a special case where the parameter space and one of the observation spaces are identical. However, separate spaces for the circuit observation and sizing parameter must be defined for the other topologies introduced below. We set the topology for the experiments to be loaded with a capacitance ($C_L = 1pF$) with unit gain, and the final sizing parameters will be rescaled based on the actual loading capacitance value.

Fig. 5 shows the agent's performance trained in the single-stage differential amplifier topology, achieving nearly optimal performance in the simple topology in Fig. 1a. The agent can minimize power consumption while meeting the hard specification (GBW).



(a) Episode reward mean



(b) I_D performance (c) GBW performance

FIGURE 5. Performance on a single-stage differential amplifier: (a) Episode reward mean, (b) Power consumption performance, and (c) Gain-bandwidth product performance.

B. TWO-STAGE DIFFERENTIAL AMPLIFIER

The second topology we examine using the proposed framework is the two-stage differential amplifier, depicted in Fig. 1b. This topology is selected to evaluate the agent's capacity to refine the circuit's sizing parameters without explicit constraints on the part of key items (such as the optimal ratio of biasing current across stages). Compared to the simple differential amplifier, the two-stage amplifier presents a more complex topology, encompassing additional state and action space dimensions. The state of the two-stage amplifier is defined as follows:

$$\begin{aligned} \mathbf{x}_t &= [I_t, \text{GBW}]^\mathsf{T} \\ \mathbf{x}^\star &= [I^\star, \text{GBW}^\star]^\mathsf{T} \\ \mathbf{q}_t &= [I_{D1}, I_{D2}]^\mathsf{T} \\ c_r &\in (-1, 1) \end{aligned} \tag{33}$$

and $I_t = 2(I_{D1} + I_{D2})$. Here, I_{Di} is sampled from the interval $[0.01, 2]$ mA, for $i = 1, 2$. The action space is two-dimensional and discrete because it includes two elements in the sizing parameter. Therefore, the action of this topology is defined as

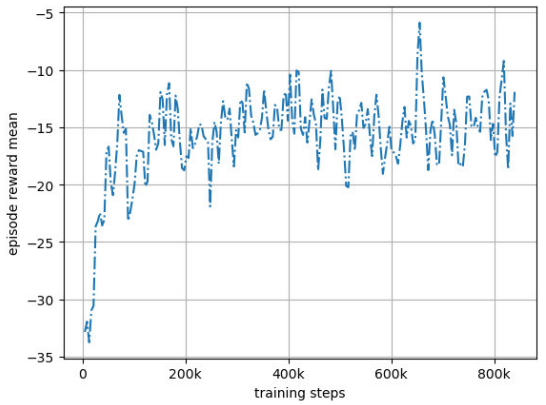
$$\mathbf{a} \in \{-1, 0, 1\}^2, \tag{34}$$

where each action denotes an adjustment to the biasing current by 0.01 mA per action. Other parameters associated with this topology remain consistent with those in the previous topology.

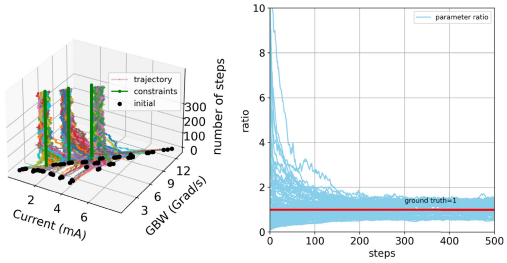
Fig. 6 shows the training and validation performance of the agent in the two-stage differential amplifier. Fig. 6a

TABLE 1. Performance comparison between fRL-AD and AutoCkt.

		AutoCkt [17]									fRL-AD (our method)								
		Gain			GBW (Grad/s)			Bias current (mA)			Gain			GBW (Grad/s)			Bias current (mA)		
Differential amplifier		min	max	avg	min	max	avg	min	max	avg	min	max	avg	min	max	avg	min	max	avg
One-stage	Desired specification		1.00			5.6			0.56			1.00			5.6			0.56	
	Designed	0.7	1.09	0.98	5.5	6.8	5.65	0.55	0.68	0.565	**1.00**	**1.00**	**1.00**	**5.5**	**5.7**	**5.65**	**0.55**	**0.57**	**0.565**
Two-stage	Desired specification		1			5.6			3.34			1			5.6			3.34	
	Designed	0.96	1.22	1.02	6.3	17.1	9.5	2.42	10.02	3.40	**1.00**	**1.00**	**1.00**	**5.30**	**5.6**	**5.33**	**3.24**	**3.40**	**3.32**



(a) Episode reward mean



(b) Agent performance (c) Parameter ratio performance

FIGURE 6. Performance on a single-stage differential amplifier: (a) Episode reward means, (b) Performance (I_D and GBW) (c) Sizing parameter ratio (I_{D2}/I_{D1}) performance.

shows the episode reward mean value per training iteration, and Fig. 6b shows the design performance according to the given design specifications (i.e., $I^\star$ and GBW*). It shows that a well-trained agent tries to fit all given specifications. Fig. 6c shows that our trained agent can figure out the nearly optimal ratio of the sizing parameters to minimize the power consumption of the given topology.

Table 1 represents the comparison of one-stage and two-stage amplifier of the proposed g_m/I_D-methodology based RL agent and the transistor level simulation-based RL agent in [17]. We aimed to design a circuit that has GAIN = 1.00, GBW = 5.6 Grad/s using a single agent with 100 independent episodes for each topology. The table provides the minimum, maximum, and average values of all 100 distinct trials. The table shows that the transistor level counterpart designed results make higher variance than the g_m/I_D-methodology-based agent.

C. OPERATIONAL TRANSCONDUCTANCE AMPLIFIER

The third circuit topology we investigate is the operational transconductance amplifier illustrated in Fig. 1c. In this case, the agent needs to account for a new design specification regarding the stability of the feedback loop. The phase margin (PM) introduces this new requirement and thus must be included in both the state space and action space of the framework.

Firstly, each vector of the state is defined as

$$\begin{aligned}\mathbf{x}_t &= [I_t, PM, \text{GBW}]^\mathsf{T}\\ \mathbf{x}^\star &= [I^\star, PM^\star, \text{GBW}^\star]^\mathsf{T}\\ \mathbf{q}_t &= [I_{D1}, I_{D2}, C_c]^\mathsf{T}\\ c_r &\in (-1, 1)\end{aligned} \tag{35}$$

and $I_t = 2 \times I_{D1} + I_{D2}$. The terms I_{Di} are sampled from the interval $[0.01, 2]$ mA for $i = 1, 2$, and C_c is sampled from $[0.01, 2]$ pF. The range of total current value I_t spans from 0 mA to 3.5 mA, which is larger than the previous cases, as more current consumption is required to meet the stability conditions. The PM spans from 0° to 180° (or 0 to 2π), and the load capacitance is $C_L = 2$ [pF].

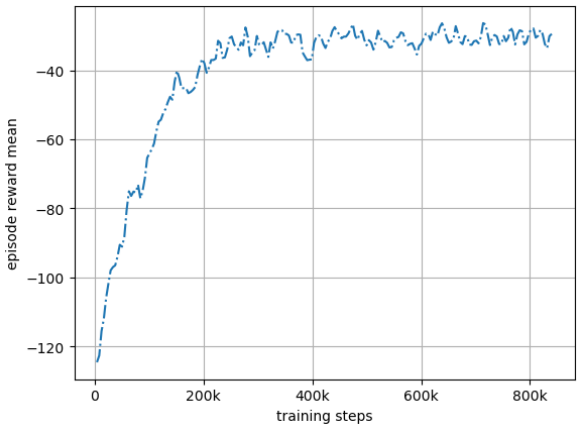
Secondly, the action space is three-dimensional and discrete, given that the sizing parameter contains three elements. The action vector for this topology is therefore expressed as:

$$\mathbf{a} \in \{-1, 0, 1\}^3. \tag{36}$$

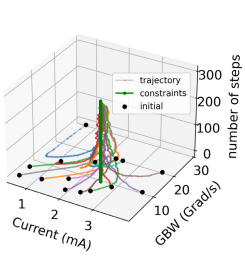
Fig. 7 shows the performance of agent training in the operational transconductance amplifier, which requires an additional design specification to stabilize the system given a feedback loop. As shown in Fig. 7a, the mean episode reward increases as the training iteration increases, which means that the agent is training to maximize the reward. Furthermore, Fig. 7b shows the design performance of the two design specifications: I_D and GBW. Along with the two specifications, the phase margin considered is shown in Fig 7c to show that our trained agent learned to suffice the PM specification while meeting the other specifications. Finally, even though the SR was not included while training since it is a DC performance, we can provide the slew rate performance results with simple computation using branch current I_D and the loaded capacitance C_L.

D. CONTINUOUS TIME LINEAR EQUALIZER

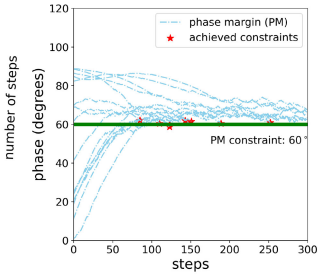
We investigate the CTLE circuit given in Fig. 1d, which involves a single-stage amplifier bridged by a resistor (R_S)



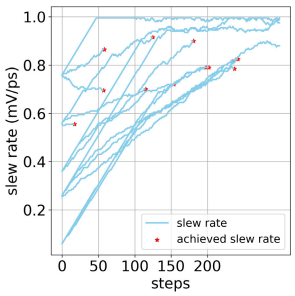
(a) Episode reward mean



(b) Agent performance



(c) Phase margin (PM) performance



(d) Slew rate (SR) performance

FIGURE 7. Performance on a single-stage differential amplifier: (a) Episode reward means, (b) Performance (I_D and GBW) (c) phase margin (PM) performance (d) slew rate (SR) performance.

and a capacitor (C_S). In this case, the agent should consider the performance that approaches the design specifications: DC level gain (A_{DC}) and the zero frequency (ω_z). Therefore, the state must include both A_{DC} and ω_z as its output $\mathbf{x}$ and its design specifications $\mathbf{x}^\star$. The agent adjusts the bridge resistance, capacitance, and branch current to get the output performance:

$$\begin{aligned} \mathbf{x}_t &= [A_{DC}, \omega_z]^\mathsf{T} \\ \mathbf{x}^\star &= [A^\star_{DC}, \omega^\star_z]^\mathsf{T} \\ \mathbf{q}_t &= [I_D, R_S, C_S]^\mathsf{T} \\ c_r &= 0 \end{aligned} \tag{37}$$

The branch current I_D is sampled from the interval $[0.01, 2]$ mA; R_S is sampled from the interval $[0, 20]$ kΩ; and C_S is sampled from $[0, 40]$pF. The zero frequency (ω_z) spans from $[0.01, 1]$ Grad/s, and the load capacitance is $C_L = 1$pF

Secondly, the action space is three-dimensional and discrete since the dimension of the sizing parameter is three. Similar to the action from the previous topology, the action is expressed as:

$$\mathbf{a} \in \{-1, 0, 1\}^3 \tag{38}$$

Fig. 8 and Table 2 show the agent's performance trained in the continuous time linear equalizer circuit. As shown in Fig. 8, the average reward of each episode while training increases as the agent steps through the environment. In this case, unlike the other topologies, we started from a fixed initial point, which eventually flattened the reward curve compared to the other topologies. However, the flatness of the training reward curve does not directly denote the stability of learning a specific topology.

Table 2 shows the results of the trained agent on the CTLE topology. As we mentioned earlier in the section, we fixed the DC gain specification to one and aimed to find the best combination of the sizing parameters to match the zero frequency. We can eventually find a policy that finds the design specifications (in this case, the zero frequency) for both 0.34 and 0.56 Grad/s, but we could find out that the design specification to approach zero frequency to 0.78 Grad/s is a hard specification that cannot be sufficed. These results show that our method can be applied to various topologies that can be formulated to (11).

TABLE 2. CTLE circuit validation results.

	design specifications	design results (variance)
DC gain	1	1 ($\pm$ 0.1)
Zero frequency (Grad/s)	0.34	0.34 ($\pm 10^{-4}$)
	0.56	0.56 ($\pm 10^{-2}$)
	0.78	0.71 ($\pm$0.05)

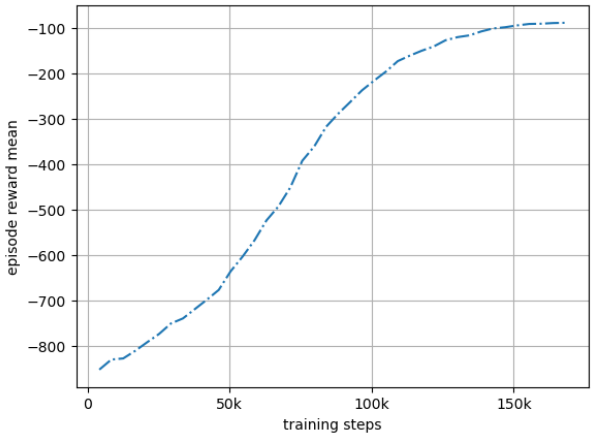


FIGURE 8. Episode reward mean value for training CTLE topology (Fig. 1d) with fixed initial points.

TABLE 3. CTLE with active inductor circuit validation results.

	design specifications	design results (variance)
DC gain	1	0.8 (± 0.1)
Zero frequency (Grad/s)	0.56	0.56 (±0.3)
	0.78	0.78 (±0.1)
Gain Gap (dB)	12.0	13.0 (±2.0)
	14.0	15.0 (±1.0)

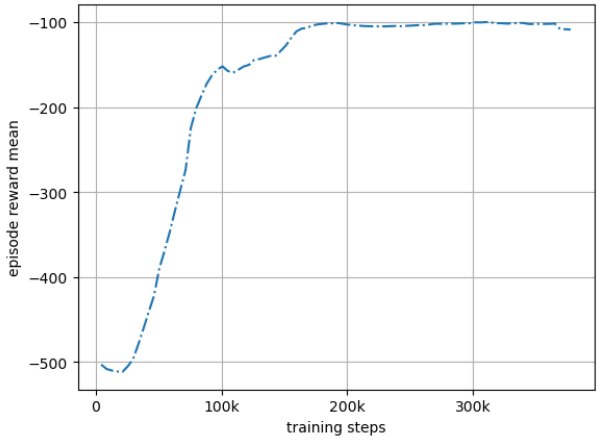


FIGURE 9. Episode reward mean value for training CTLE with active inductor topology (Fig. 1e) from fixed initial points.

E. CONTINUOUS TIME LINEAR EQUALIZER WITH ACTIVE INDUCTOR

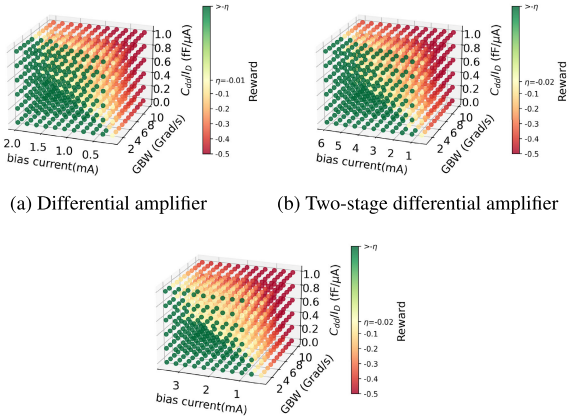
We train the CTLE circuit given in Fig. 1e, which includes active inductive loads on each differential pair. This enhances the bandwidth by adding an inductive peak at Nyquist frequency [22]. The topology includes R_S and C_S degenerations between the differential pairs. In this topology, the agent should consider all the performances in the CTLE topology as in Fig. 1d: A_{DC} and ω_z. Furthermore, we include a new design specification: the gap between the peak and DC gain(G) in the dB scale. According to the state, action, and reward formulation, each component of the state $\mathbf{s}_t$ is

$$\begin{aligned}\mathbf{x}_t &= [A_{DC}, \omega_z, G]^\mathsf{T},\\ \mathbf{x}^\star &= [A_{DC}^\star, \omega_z^\star, G^\star]^\mathsf{T},\\ \mathbf{q}_t &= [I_D, R_S, C_S]^\mathsf{T},\\ c_r &= 0.\end{aligned} \tag{39}$$

Assuming $C_L = 2$ pF, and $g_m/I_D = 10$, the branch current I_D is sampled from the interval $[0.1, 20]$ mA. R_S is sampled from the interval $[0.1, 20]$ kΩ, and C_S is sampled from $[0, 20]$ pF. The DC gain is aimed at approaching 1 (or 0 dB), where the zero frequency (ω_z) and gain gap (G) specifications lie inside the interval $[0.01, 1]$ Grad/s and $[0, 20]$ dB respectively.

The action space is a three-dimensional discrete vector following the dimension of the parameter vector $\mathbf{q}_t$, expressed as

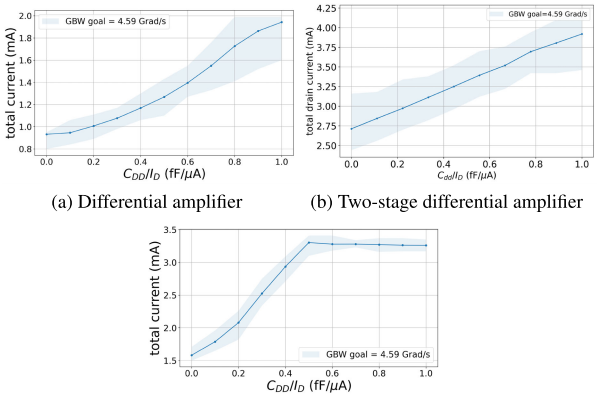
$$\mathbf{a} = \{-1, 0, 1\}^3. \tag{40}$$



(a) Differential amplifier (b) Two-stage differential amplifier

(c) Operational transconductance amplifier

FIGURE 10. Three-dimensional success maps with three topologies: Differential amplifier, two-stage differential amplifier, and operational transconductance amplifier.



(a) Differential amplifier (b) Two-stage differential amplifier

(c) Operational transconductance amplifier

FIGURE 11. Bias current with respect to C_{dd}/I_D with fixed GBW specifications.

Here, the reward functions also follow the distance measure according to the type of the given design specifications.

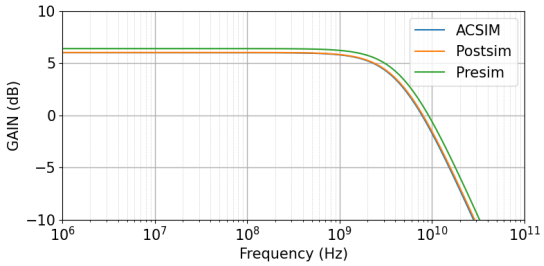
Fig. 9 and Table 3 represent the training procedure and the validation results of the agent on the topology with a fixed initial parameter vector. As shown in Fig. 9, the average reward gradually increases throughout the training process. Table 3 shows the results of the fully trained agent on the topology. We can conclude that the trained agent can find the nearly optimal combination of the given design specifications. These results show that our method can be generalized to state-of-the-art topologies formulated to (11).

F. CONSIDERING ON DEVICE PARASITICS

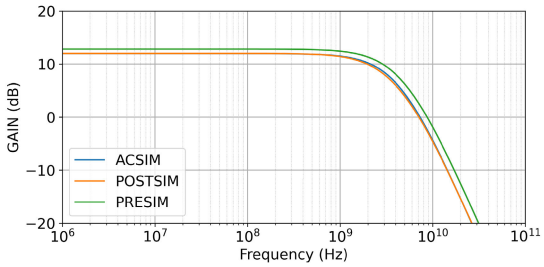
Fig. 10 and Fig. 11 represent the empirical results of the agent trained in a setting where parasitic capacitance exists in amplifier topologies. As aforementioned, we did not expose the exact value of the parasitic capacitance to the training agent. Instead, we trained the agent to consider the relationship between the input and output relationship to take

TABLE 4. pre-layout simulation, post-layout simulation comparison between AC equivalent results with device parasitic.

Topology	pre-layout simulation		post-layout simulation		Our method	
	Gain	GBW	Gain	GBW	Gain	GBW
One-stage differential amplifier (Fig. 1a)	2.09	10.39	2.003	9.181	1.998	8.970
Two-stage differential amplifier (Fig. 1b)	4.381	13.06	3.983	9.957	3.968	10.804



(a) Differential amplifier



(b) Two-stage differential amplifier

FIGURE 12. Frequency response of pre-layout simulation, post-layout simulation, AC simulation results.

action and make valid decisions (i.e., allow higher power consumption to drive the load capacitance).

Fig. 10 shows the success map of three amplifier topologies, where the horizontal axis represents the desired specifications: $I^{\star}$ and GBW*. The vertical axis represents the normalized parasitic capacitance (C_{dd}/I_D or C_{gg}/I_D). The region is categorized into two regions: the *success* region and the *failure* region. The two regions are classified based on the highest reward obtained during the episode. As shown in Fig. 10, the success region reduces as the normalized parasitic (C_{dd}/I_D) of the layout increases.

Fig. 11 shows the total current of the topology with fixed GBW specifications to different device parasitic. The horizontal axis refers to the normalized device parasitic, and the vertical axis represents the total bias current. The figure shows that our agent designs to use more currents to drive the same load capacitance (C_L), using only the information of the relation between states without any additional knowledge given. This additional consumption of current shows that our trained agent can consider unidentifiable uncertainties using the relationship between elements.

Table 4 shows the numerical results of the pre-layout simulation and post-layout simulation results compared to our AC equivalent circuit results with g_m/I_D in a 40 nm manufacturing process. Note that the process was not considered in the training phase. We conduct respective simulations for the one-stage and two-stage fully differential amplifiers. For one-stage differential amplifier under device parasitics $C_{dd}/I_D = 0.0188$ ($fF/\mu A$) and $C_{gg}/I_D = 0.0138$ ($fF/\mu A$), the pre-layout simulation with $C_L = 100$ fF, $I_D = 0.605$ mA returns GAIN = 2.09, GBW = 10.39 GHz. In the identical setting, the post-layout simulation results are GAIN = 2.003 and GBW = 9.181 GHz, and the AC equivalent results are GAIN = 1.998 and GBW = 8.970 GHz. For a two-stage differential amplifier with $C_1 = 100$ fF, $C_2 = 200$ fF, $I_{D1} = 0.605$ mA and $I_{D2} = 1.21$ mA, the pre-layout simulation results are GAIN = 4.381 and GBW = 13.06 GHz. The post-layout simulation results are GAIN = 3.983 and GBW = 9.957 GHz, and the AC equivalent results are GAIN = 3.968 and GBW = 10.804 GHz. Fig. 12 shows the frequency response curves of the pre-layout simulation, post-layout simulation, and our AC simulation results. These results show that our methodology can effectively capture the post-layout effect while securing process independence.

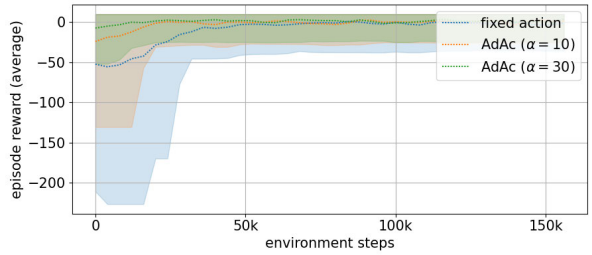
G. ADAPTIVE ACTION TECHNIQUE

1) TRAINING

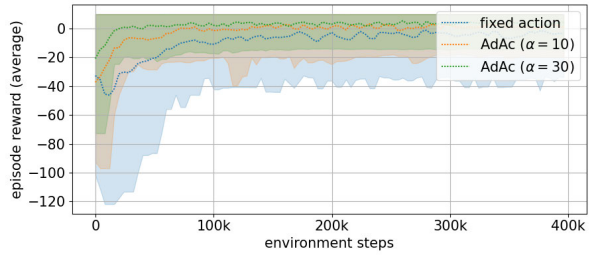
In order to check the sample efficiency of the proposed *adaptive action* technique, we run through different agents with different adaptation parameters (α). Fig. 13 shows the average rewards for a given amount of episodes. We trained 7 parallel agents with identical design goals and measured the maximum, average, and minimum value of the episode reward. The horizontal axis in the figure represents the total number of steps during the training phase, and the vertical axis represents the episode reward. The dotted line shows the average value of the episode reward. The vivid colors filling the graph represent the variance of each episode reward. The maximum value of the vivid region is the maximum episode reward returned by the environment while training, and the minimum value is the minimum episode reward returned by the environment. We can see that for all topologies, increasing the adaptation parameter α to higher values (0, 10, 30) accelerates the training of the current environment. Fig. 13 shows the difference between frameworks in simple differential amplifiers. for training differential amplifiers with fixed action frameworks, the average episode reward crosses the zero line after approximately 70-80K steps, where increasing α to 10 and 30 reduces the number to approximately 10K and 30K steps.

2) VALIDATION

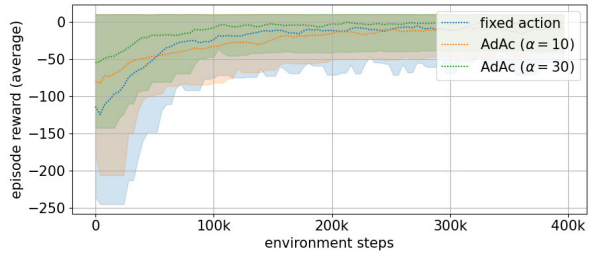
The *adaptive action* technique for the environments is effective not only on the newly trained agents shown in



(a) Differential amplifier

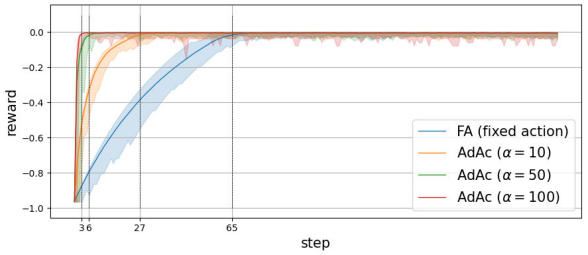


(b) Two-stage differential amplifier

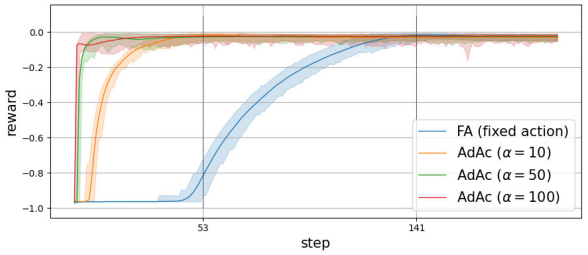


(c) Operational transconductance amplifier

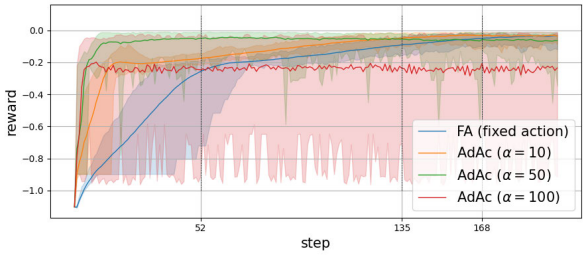
FIGURE 13. Training rewards: episode reward mean with respect to environment steps. *Adaptive action* is conducted with different adaptation parameter values (α).



(a) Differential amplifier



(b) Two-stage differential amplifier



(c) Operational transconductance amplifier

FIGURE 14. Validation rewards: reward per step during the validation phase. Meaningful acceleration exists in all (a) single-stage differential amplifiers, (b) two-stage differential amplifiers, and (c) operational transconductance amplifiers.

Fig. 13, but it is also effective on pre-trained agents using *fixed action* environments. To show the effectiveness of our proposed environment framework, we conduct additional experiments with agents trained in *fixed action* environments and validate the pre-trained agent with *adaptive action* environment for validation.

The results are shown in Fig. 14, where the horizontal axis represents the number of steps in a single episode, and the vertical axis represents the step reward ($r(s_t, a_t)$). The bold line represents the average step reward along 100 episodes, starting from the same point, and the vivid region represents the variation of each episode. This figure also shows the convergence time of a moderately trained agent. It is empirically shown that the agent nearly converges to a moderate reward level under the steps computed in section IV-D, falling under the total steps of NK. For example, the convergence time lies under KN for the fixed action validation curve. In Fig. 14a, the convergence time lies under 200 ($K = 200, N = 1$), and in Fig. 14b, it lies under 400 ($K = 200, N = 2$). Similarly, in Fig. 14c, the convergence time lies under 600 ($K = 200, N = 3$).

As shown in Fig. 14, increasing the adaptive parameter α from 0 to 100 can converge the design process faster than the fixed action environment. The aftereffect is insignificant since the topology is relatively simple compared to the others. However, for more complex topologies such as two-stage amplifiers and operational transconductance amplifiers, increasing the adaptive parameter can escalate the variance of the design results, which deteriorates stable convergence. As shown in Fig. 14b and Fig. 14c, the reward slope increases as the parameter increases. At the same time, the variance of each episode also grows, which makes it harder for the agent to converge into a given reward threshold value. Therefore, picking an adequate value of the adaptive parameter can balance the agent between convergence speed and stability.

VI. CONCLUSION

In this paper, we proposed a novel analog circuit designing automation method that optimizes parameters with an RL framework. The agent considers the AC characteristics with fixed DC operation points by applying the g_m/I_D methodology as a baseline knowledge. Our method provides a stable, robust agent without applying additional learning tools. In addition, we formulated a unified mathematical formulation of an optimization problem for analog circuit design to optimize performance. We proposed a sequential

framework where each subproblem can be solved with the pre-trained agent above. We also proposed an environmental framework called the *adaptive action space* for the environment. We have shown with various experiments that *adaptive action space* are faster compared to the fixed action counterpart, and experiments showed that an agent trained with a non-adaptive space can easily perform well in the new *adaptive action space* applied environment without any retraining. This can boost the speed of running simulations as well as the reduction in costs for running simulators. However, since our methodology requires a small-signal equivalent circuit, it poses errors when non-linear components are approximated into linear components, eventually causing additional uncertainties. Therefore, our method cannot be tested through non-linear and switched capacitance circuits, which limits its applications. Furthermore, the objective of designing a circuit is often considered multi-objective optimization, introducing unavoidable limitations to current scalar rewards. Therefore, future studies could extend the current research by exploiting novel RL frameworks for environmental uncertainties and exploring adequate vector reward functions. Another interesting future work would consider additional performance parameters such as noise and PSRR for specific applications.

REFERENCES

[1] 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 *Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*, Nov. 2013, pp. 74–81.

[2] P. Jespers, *The g_m/I_D Methodology, a Sizing Tool for Low-Voltage Analog CMOS Circuits: The Semi-Empirical and Compact Model Approaches*. New York, NY, USA: Springer, 2009.

[3] J. Kim, R. Jhaveri, J. Woo, and C.-K. Ken Yang, "Device-circuit co-optimization for mixed-mode circuit design via geometric programming," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2007, pp. 470–475.

[4] J. Kim, J. Lee, L. Vandenberghe, and C.-K. Ken Yang, "Techniques for improving the accuracy of geometric-programming based analog circuit design optimization," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, Nov. 2004, pp. 863–870.

[5] J. Kim, L. Vandenberghe, and C. K. Yang, "Convex piecewise-linear modeling method for circuit optimization via geometric programming," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 11, pp. 1823–1827, Nov. 2010.

[6] M. del Mar Hershenson, "CMOS analog circuit design via geometric programming," in *Proc. Amer. Control Conf.*, Jun. 2004, pp. 3266–3271.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[8] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[9] B. Liu, D. Zhao, P. Reynaert, and G. G. E. Gielen, "GASPAD: A general and efficient mm-wave integrated circuit synthesis method based on surrogate model assisted evolutionary algorithm," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 2, pp. 169–182, Feb. 2014.

[10] W. Lyu, P. Xue, F. Yang, C. Yan, Z. Hong, X. Zeng, and D. Zhou, "An efficient Bayesian optimization approach for automated optimization of analog circuits," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 6, pp. 1954–1967, Jun. 2018.

[11] 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 *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3306–3314.

[12] S. Zhang, F. Yang, C. Yan, D. Zhou, and X. Zeng, "An efficient batch-constrained Bayesian optimization approach for analog circuit synthesis via multiobjective acquisition ensemble," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 1, pp. 1–14, Jan. 2022.

[13] 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 *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, Dec. 2021, pp. 1219–1224.

[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 *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.

[15] K. Hakhamaneshi, M. Nassar, M. Phielipp, P. Abbeel, and V. Stojanovic, "Pretraining graph neural networks for few-shot analog circuit modeling and design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 7, pp. 2163–2173, Jul. 2023.

[16] C. Vişan, O. Pascu, M. Stănescu, E.-D. Şandru, C. Diaconu, A. Buzo, G. Pelz, and H. Cucu, "Automated circuit sizing with multi-objective optimization based on differential evolution and Bayesian inference," *Knowl.-Based Syst.*, vol. 258, Dec. 2022, Art. no. 109987.

[17] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolic, "AutoCkt: Deep reinforcement learning of analog circuit designs," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 490–495.

[18] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *Eur. J. Oper. Res.*, vol. 290, no. 2, pp. 405–421, Apr. 2021.

[19] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," 2016, *arXiv:1611.09940*.

[20] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[21] M. Choi, Y. Choi, K. Lee, and S. Kang, "Reinforcement learning-based analog circuit optimizer using gm/ID for sizing," in *Proc. 60th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2023, pp. 1–6.

[22] D. Thulasiraman, G. Chiranjeevi, J. S. Gaggatur, and K. S. S. Reddy, "A 18.6 fJ/bit/dB power efficient active inductor-based CTLE for 20 Gb/s high speed serial link," in *Proc. IEEE Int. Conf. Electron., Comput. Commun. Technol. (CONECCT)*, Jul. 2019, pp. 1–6.

[23] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, Dec. 2018.

[24] B. Razavi, *Fundamentals of Microelectronics*. Hoboken, NJ, USA: Wiley, 2021.

[25] F. Silveira, D. Flandre, and P. G. A. Jespers, "A g_m/I_D based methodology for the design of CMOS analog circuits and its application to the synthesis of a silicon-on-insulator micropower OTA," *IEEE J. Solid-State Circuits*, vol. 31, no. 9, pp. 1314–1319, Sep. 1996.

[26] P. G. Jespers and B. Murmann, *Systematic Design of Analog CMOS Circuits*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[27] E. Tlelo-Cuautle and A. C. Sanabria-Borbon, "Optimising operational amplifiers by evolutionary algorithms and g_m/I_D method," *Int. J. Electron.*, vol. 103, no. 10, pp. 1665–1684, Oct. 2016.

[28] K. Miettinen, *Nonlinear Multiobjective Optimization*, vol. 12. New York, NY, USA: Springer, 1999.

[29] H. Wang, J. Yang, H.-S. Lee, and S. Han, "Learning to design circuits," 2018, *arXiv:1812.02734*.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 8540, pp. 529–533, Feb. 2015.

[31] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[32] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

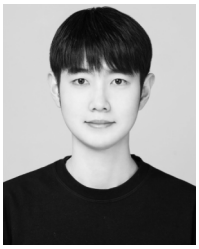
[33] P. Christodoulou, "Soft actor-critic for discrete action settings," 2019, *arXiv:1910.07207*.

[34] R. A. Howard, *Dynamic Programming and Markov Processes*. Cambridge, MA, USA: MIT Press, 1960.

SUNGWEON HONG (Graduate Student Member, IEEE) received the B.S. degree in electronic engineering from Ajou University, Suwon, South Korea, in 2021. He is currently pursuing the combined master's and Ph.D. degree with Hanyang University, Seoul, South Korea. His research interests include reinforcement/machine learning, optimization for wireless communications, and statistical signal processing.

YUNSEOB TAE (Graduate Student Member, IEEE) received the B.S. degree in electronics engineering from Hanyang University, South Korea, in 2022, where he is currently pursuing the Ph.D. degree in electronics engineering. His primary research interests include machine learning for wireless communication and indoor localization.

DONGJUN LEE (Graduate Student Member, IEEE) received the B.S. degree in electronic engineering from Jeju National University, Jeju, South Korea, and the M.S. degree in electrical engineering from Hanyang University, Seoul, South Korea, in 2020, where he is currently pursuing the Ph.D. degree. His research interests include high-speed analog and mixed-signal (AMS) circuit design and automation.

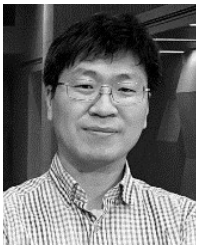


GIJIN PARK (Graduate Student Member, IEEE) received the B.S. degree in electronic engineering from Hanyang University, Seoul, South Korea, in 2022. He is currently pursuing the Ph.D. degree in electrical engineering with Texas A&M University, College Station, TX, USA. His research interests include high-speed analog, mixed-signal (AMS) circuit design, and analog-to-digital converters (ADCs).

JAEMYUNG LIM (Member, IEEE) received the bachelor's degree in electrical and computer engineering from Hanyang University, Seoul, South Korea, in 2011, and the M.S. and Ph.D. degrees in electrical and computer engineering from Georgia Tech, Atlanta, GA, USA, in 2013 and 2017, respectively.

From 2004 to 2007, he was a Senior Circuit Designer at Apple Inc., Cupertino, CA, USA. He is currently an Assistant Professor of electronic engineering with Hanyang University. His scholarly pursuits revolve around various areas, including display driver ICs for OLEDoS and LEDoS, ultrasound imaging systems, and power management IC design. Furthermore, he harbors a keen interest in leveraging deep learning and generative AI for design methodology and automation.

KYUNGJUN CHO (Member, IEEE) received the B.S. degree in electronic engineering from Ajou University, Suwon, South Korea, in 2014, and the M.S. and Ph.D. degrees in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2016 and 2019, respectively. In 2019, he joined SK hynix Semiconductor Inc., Icheon, South Korea, where he has been involved in DRAM circuit design. His research interests include low-power and high bandwidth 3-D DRAM circuit design, programmable memory built-in self-test (PMBIST), electromagnetic compatibility (EMC), and signal and power integrity (SI/PI).



CHUNSEOK JEONG (Member, IEEE) received the B.S. degree in electronics engineering from the University of Seoul, in 2003, and the M.S. degree in electrical and computer engineering from Hanyang University, in 2005. In 2005, he joined SK hynix Semiconductor Inc., South Korea, and participated in the design works for the development of high-bandwidth memory (HBM), including DDR2, DDR3, DDR4 managed DRAM, and processing in memory (PIM). He is currently in charge of HBM product development. His research interests include design of high-bandwidth low-power memory and PIM system applications.

MYEONG-JAE PARK (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Seoul National University, Seoul, South Korea, in 2003, 2006, and 2014, respectively. He is currently in charge of the HBM Design Division SK hynix Inc., Icheon, South Korea, as a Vice President, where his research interests include 3D-DRAM, low-power mixed signal systems, and their design methodologies. Prior to joining SK hynix Inc., in 2014, he was a Principal Engineer, from 2004 to 2009, with Anapass, South Korea, involved in LCD intra-panel interfaces.

SONGNAM HONG (Member, IEEE) received the B.Sc. and M.Sc. degrees in electrical and computer engineering from Hanyang University, Seoul, South Korea, in 2003 and 2005, respectively, and the Ph.D. degree in electrical engineering from the University of Southern California (USC), Los Angeles, CA, in 2014. From 2005 to 2009, he was a Research Engineer with the Telecommunication Research and Development Center, Samsung Electronics, Suwon, South Korea, and from 2014 to 2016, he was a Senior Research Engineer with the Ericsson Research, San Jose, CA, USA. From 2016 to 2020, he was an Assistant Professor with Ajou University, Suwon. He is currently an Associate Professor with Hanyang University. His main research interests include the areas of large-scale optimization, machine learning, reinforcement learning, information theory, and statistical signal processing.

JAEDUK HAN (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Seoul National University, Seoul, South Korea, in 2007 and 2009, respectively, and the Ph.D. degree in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 2017.

He is currently an Assistant Professor of electronic engineering with Hanyang University, Seoul. He has held various internship and full-time positions at TLI, Seongnam, South Korea; Altera, San Jose, CA, USA; Intel, Hillsboro, OR, USA; Xilinx, San Jose, CA, USA; and Apple, Cupertino, CA, USA, where he worked on digital, analog, and mixed-signal integrated circuit designs and design automation. His research interests include high-speed analog and mixed-signal (AMS) circuit design and automation.