Multi-Agent Reinforcement Learning in Designing the Low-Dropout Regulator Circuits

THANG QUOC NGUYEN, Department of Electrical and Computer Engineering, Faculty of Engineering and Applied Science, Memorial University of Newfoundland, Canada

LIHONG ZHANG, Department of Electrical and Computer Engineering, Faculty of Engineering and Applied Science, Memorial University of Newfoundland, Canada

OCTAVIA A. DOBRE, Department of Electrical and Computer Engineering, Faculty of Engineering and Applied Science, Memorial University of Newfoundland, Canada

TRANG HOANG, Department of Electronics, Faculty of Electrical and Electronics Engineering, Ho Chi Minh City University of Technology (HCMUT), Vietnam and Vietnam National University Ho Chi Minh City (VNU-HCM), Vietnam

TRUNG Q. DUONG, Department of Electrical and Computer Engineering, Faculty of Engineering and Applied Science, Memorial University of Newfoundland, Canada and School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, UK

Low-dropout regulator (LDO) circuit, whose function is to provide a power supply robust to variations in process, voltage, and temperature (PVT), is an essential part in any system-on-chip. Therefore, the design of this circuit must satisfy various intricate specifications, rendering the design process generally perceived as tedious and lengthy. As a result, previous research has been conducted to explore the use of machine learning, particularly reinforcement learning, in speeding up and automating the LDO design process, especially the sizing phase. However, the results of these works are limited in terms of the number of design variables and specifications handled by the automation engine. This study presents the application of single-agent proximal policy optimization (PPO) and multi-agent proximal policy optimization (MAPPO), including both parameter-separated and parameter-sharing methods, to address the LDO sizing automation problem. The experimental result shows that the PPO- and MAPPO-based implementation in

Corresponding author is Trang Hoang (hoangtrang@hcmut.edu.vn).

This paper has been accepted in part for presentation at the IEEE International Symposium on Circuits and Systems (IEEE ISCAS 2025), London, UK, May 2025.

Authors' Contact Information: Thang Quoc Nguyen, nqthang@mun.ca, Department of Electrical and Computer Engineering, Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. John's, NL, Canada; Lihong Zhang, lzhang@mun.ca, Department of Electrical and Computer Engineering, Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. John's, NL, Canada; Octavia A. Dobre, odobre@mun.ca, Department of Electrical and Computer Engineering, Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. John's, NL, Canada; Trang Hoang, hoangtrang@hcmut.edu.vn, Department of Electronics, Faculty of Electrical and Electronics Engineering, Ho Chi Minh City University of Technology (HCMUT), Dien Hong Ward, Ho Chi Minh City, Vietnam and Vietnam National University Ho Chi Minh City (VNU-HCM), Linh Xuan Ward, Ho Chi Minh City, Vietnam; Trung Q. Duong, tduong@mun.ca, Department of Electrical and Computer Engineering, Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. John's, NL, Canada and School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast, UK.

LDO sizing automation outperforms that of the classical particle swarm optimization algorithm. We demonstrate that the parameter-separated MAPPO features the most effective learning process compared with other PPO-based benchmarks, resulting in a design result that is competitive to that of a well-known commercial tool.

CCS Concepts: • **Hardware → Methodologies for EDA**; • **Computing methodologies → Reinforcement learning**.

Additional Key Words and Phrases: Bandgap reference circuit, computer-aided design (CAD), multi-agent proximal policy optimization (MAPPO), low-dropout regulator (LDO), multi-agent reinforcement learning (MARL).

ACM Reference Format:
Thang Quoc Nguyen, Lihong Zhang, Octavia A. Dobre, Trang Hoang, and Trung Q. Duong. 2025. Multi-Agent Reinforcement Learning in Designing the Low-Dropout Regulator Circuits. 1, 1 (November 2025), 21 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The low-dropout regulator (LDO) is essential for power management units, which are used to ensure a robust and minimal noise power supply to different blocks within the integrated circuit (IC) [1–3]. The ideal LDO must have the output voltage that is robust against process, voltage, temperature (PVT), and load variations, as well as the dropout voltage should be maintained as small as possible [4, 5]. Due to these characteristics, LDO is widely used in various real-world applications, such as power management IC in energy harvesting chips [6], biomedical sensors [7–9], optical devices [10], Internet-of-Thing circuits[11–13], and satellite systems [14].

The design process of the LDO, like other analog ICs, can be generally divided into three principal phases: (1) topology design, during which the circuit structure is formed [15]; (2) sizing, where the physical properties of each element in the circuit, such as bias voltage, width, and length of transistor, are determined [16]; and (3) layout, in which the physical placement and routing are performed [17, 18]. In the sizing phase, calculating the physical properties to make the entire circuit achieve the predetermined specifications is considered as a time-consuming task. Due to the inherent nonlinearity between physical properties and electrical performances, as well as the complex trade-off among these performances, designers often experience a tedious iterative trial-and-error process of fine-tuning their initial sizing calculation until reaching the predefined specifications [19]. To avoid performing a lengthy manual trial-and-error sizing process, design automation has been proposed. To avoid performing a lengthy manual trial-and-error sizing process, design automation has been proposed. The simplest design automation approach is to divide the design domain into several blocks and write a script to sweep and run the simulation at each block to find the optimal settings where the constraints and design rules are satisfied. However, searching across the whole domain without any holistic coordination will require a large amount of time and resources, since it is nontrivial to evaluate all the specifications. When it comes to PVT-variation consideration, this time-consuming simulation process has to be repeated for every PVT corner, thus significantly degrading the optimization accuracy and efficiency.

To avoid such a large amount of time and computation consumption, several more strategic sizing automation approaches have been proposed. So far, precomputed lookup table-based methods were introduced to address the sizing automation problem in designing the bandgap reference (BGR) [20, 21] and amplifier [21–23] circuits. Besides providing a systematic way to address the sizing automation problem with accurate results, these precomputed lookup table-based methods are difficult to generalize into a general design automation framework, since the lookup table is generated from simulation data of a certain CMOS technology node. Moreover, the analytical lookup strategies used in the precomputed lookup table-based sizing automation methods are only useful for these specific design problems, and their accuracy largely depends on the density of the data collected. Therefore, simulation-based design automation tools (e.g, the

Cadence Virtuoso Analog Design Environment (ADE) GXL), are preferred by the designers. In those commercial design automation tools, the circuit performance is derived from netlist simulation rather than any pre-generated dataset. Although having powerful search capabilities and being able to be applied to any circuits without requiring any dataset, these tools are licensed and typically highly expensive. Even though some academic tools feature academic package or student version, certain advanced modules may not be included or circuit size is largely constrained. Therefore, those commercial tools or their advanced modules are too cost-prohibitive for regular industrial designers or academic researchers who do not have substantial financial support to access. Moreover, the EDA vendors normally publicize no detailed algorithms behind their optimization tools, and the users cannot fine-tune the hyperparameters controlling the behavior of the tools, making it difficult for them to improve the performance of these tools. Due to the cost-prohibition and opacity of the commercial advanced sizing tools, various simulation-based automation frameworks based on the use of heuristic algorithms [24–26] and Bayesian optimization (BO) [27, 28] have been proposed. In these simulation-based frameworks, which also rely on netlist simulations (similar to the Cadence sizing tool), an algorithm that has successfully run for a specific circuit can be easily employed on another circuit just by replacing the netlist file called in each optimization iteration. Additionally, since the simulation-based design automation method allows calling the circuit simulator in each optimization iteration, the output results are more accurate than the precomputed lookup table-based method. Moreover, the mechanism behind these academia-developed simulation-based frameworks is transparent, making it possible for other users to learn and further improve. However, the use of heuristic algorithms and BO faces some limitations when the design problem becomes more complex: heuristic search could easily get stuck in local optima when dealing with a high-dimensional design space, while the BO's computational cost scales cubically with the number of samples, potentially limiting its applicability in large-scale problems [29, 30]. Additionally, BO's sampling efficiency will degrade if the number of design variables increases [19].

In recent years, reinforcement learning (RL) has gained attention as a promising tool for addressing the analog IC design automation problem [29, 31–34]. Reference [32] shows that the performance of the circuit sizing task handled by RL outperforms that of BO and heuristic algorithms. This result is explainable because RL employs neural networks to model how to select an action to maximize the optimization targets, while the BO uses the Gaussian distribution (which is less efficient than the neural network), and the heuristic algorithm does not have any mechanism to model the relationship between the design variables and design goals. References [32, 34] specifically investigated the application of the deep deterministic policy gradient (DDPG) RL method in LDO design automation. In these studies, the reference voltage input to the LDO circuit is assumed to be provided by a constant voltage source. In real-world design, this input reference voltage comes from the output of the BGR circuit [4, 35, 36], which varies with PVT. To make the design automation flow closer to the manual one and take the non-ideality of the BGR's output into the LDO design consideration, in this work, the LDO design automation problem is widened by including the design of the BGR circuit. However, letting one RL agent handle a high-dimensional environment with numerous action variables and a large observation space could degrade its efficiency, since there is too much information needed to be modelled by just one neural network.

To avoid the curse of dimensionality problem that may be encountered when applying just one RL agent to a large-scale problem, in this work we propose to utilize the multi-agent RL framework to address the LDO design automation problem. In our proposed multi-agent RL implementation, the action space and observation space are divided into several sub-regions. Multiple neural networks are employed to learn the data collected from the corresponding sub-region, which can reduce the amount of training workload on one neural network. The effectiveness of the multi-agent proximal policy optimization (MAPPO) algorithm in handling benchmark scenarios where agents not only act individually but

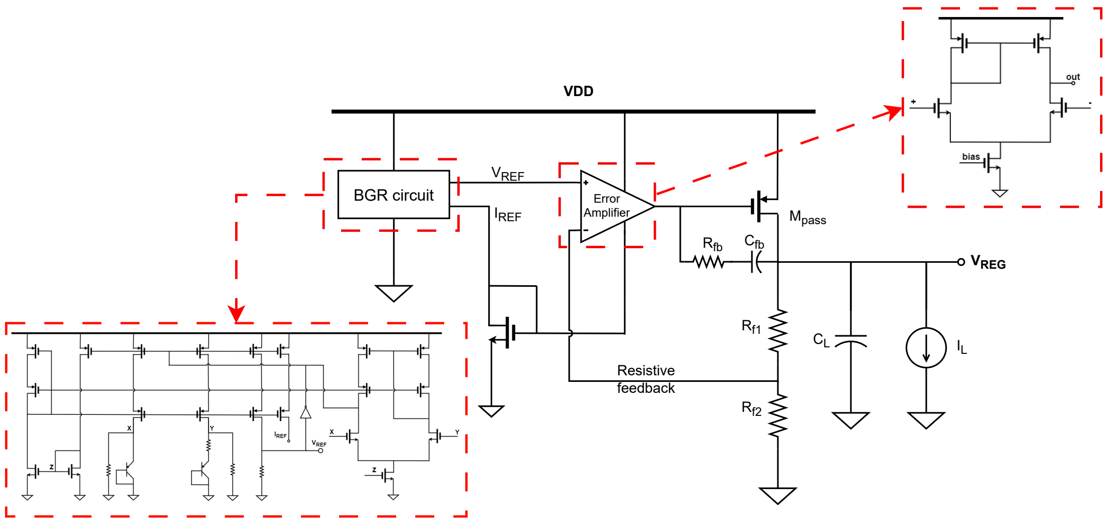


Fig. 1. Schematic of the LDO circuit.

also cooperate together has been demonstrated in previous work [37]. In the realm of analog circuit design, the prowess of multi-agent RL has been investigated on sizing automation for amplifiers [38, 39]. In the LDO design automation problem, it is observed that the BGR's output voltage affects the accuracy of the whole LDO circuit; on the other hand, the performance of BGR circuit does not depend on the other parts of the LDO. Therefore, we employ the MAPPO algorithm to independently learn to design the BGR and other parts of the LDO while keeping them cooperating together to let the whole LDO reach a good performance.

The contributions of this paper can be summarized as follows:

(1) Proposes the use of the multi-agent RL framework in tackling the LDO sizing automation task, which could facilitate the breakdown of challenging optimization tasks.
(2) Utilizes MAPPO with both parameter-separated and parameter-sharing configurations to address the LDO sizing automation problem.
(3) Provides an analysis comparing the results from the implementation of the single-agent PPO, the MAPPO with parameter-separated and parameter-sharing settings in terms of learning stability and exploration ability.
(4) Compares the results from implementing the proposed PPO-based algorithms with particle swarm optimization and the Cadence Virtuoso ADE GXL tool, and discusses the benefits of using the proposed framework over these two alternatives.

The structure of this paper is organized in the following manner: Section 2 provides an introduction to LDO, PPO and MAPPO. Section 3 explains the formulation of the optimization problem, as well as the optimization process. Results are presented, compared and discussed in Section 4. Finally, Section 5 concludes the paper.

Table 1. Specifications for the LDO circuit.

No.	Metrics	Notation	Constraint
1	Dropout voltage	V_d	$\leq 220\ mV$
2	Load regulation	V_L	$\leq 36\ mV$
3	PSRR at 1 kHz, heavy-load	$PSRR_{1kHz,HL}$	$\leq -30\ dB$
4	PSRR at 1 kHz, light-load	$PSRR_{1kHz,LL}$	$\leq -30\ dB$
5	PSRR at 1 MHz, heavy-load	$PSRR_{1MHz,HL}$	$\leq -20\ dB$
6	PSRR at 1 MHz, light-load	$PSRR_{1MHz,LL}$	$\leq -20\ dB$
7	Phase margin at heavy-load	PM_{HL}	$\geq 60^\circ$
8	Phase margin at light-load	PM_{LL}	$\geq 60^\circ$
9	Quiescent current	I_q	$\leq 200\ \mu A$
10	BGR's temperature coefficient	TC	$\leq 10\ ppm/^\circ C$
11	BGR'S output voltage	V_{REF}	$= 0.9 \pm 0.02\ V$
12	BGR's loop gain	LG_{BGR}	$\geq 40\ dB$
13	BGR's phase margin	PM_{BGR}	$\geq 60^\circ$

2 Preliminary

2.1 Low-Dropout Regulator (LDO)

Fig. 1 depicts a typical structure of LDO circuit. It consists of three main sub-circuits: a BGR circuit, an error amplifier, and a pass transistor. The BGR circuit is to produce a reference voltage V_{REF} which is robust against PVT variation. Output voltage is maintained stable by using negative feedback formed by the error amplifier and the pass transistor. Under steady-state conditions, the LDO exhibits a linear relationship between the input and output voltages, effectively acting as a resistor. This relationship is expressed as

$$V_{REG} \approx \left(1 + \frac{R_{fb1}}{R_{fb2}}\right) V_{REF}, \tag{1}$$

where V_{REG} is the output voltage of the LDO and V_{REF} is the reference voltage provided by the BGR circuit. This equation highlights the LDO's ability to maintain a consistent output voltage despite variations in the supply voltage, making it a crucial component in power management systems requiring precise voltage regulation.

Table 1 summarizes the specifications for the LDO circuit that will be designed in this work. Rows 1 to 9 are the performance metrics of the whole LDO system, including dropout voltage (which is the difference between supply voltage and LDO's output voltage), load regulation, power supply rejection ratio (PSRR) and phase margin measured in both heavy-load (10 mA) and light-load (10 μA) modes, and quiescent current. The last four rows list the performance metrics of the BGR subcircuit. The metric of temperature coefficient, which characterizes the variations of BGR's output voltage, is calculated as

$$TC(ppm/^\circ C) = \frac{V_{REF,max} - V_{REF,min}}{V_{REF,norm} \times (T_{max} - T_{min})} \times 10^6, \tag{2}$$

where $V_{REF,max}$ and $V_{REF,min}$ are the maximum and minimum value of V_{REF} during its temperature sweeping simulation, respectively, $V_{REF,norm}$ is the value of V_{REF} at 25°C, $T_{max} = 125^\circ C$, $T_{min} = -40^\circ C$.

2.2 Proximal Policy Optimization (PPO)

In this study, the PPO algorithm (with single-agent and multi-agent settings) is utilized as the engine within our design automation framework. PPO is a policy-based actor-critic RL algorithm, designed to address the issues of data efficiency and robustness, which are key limitations of previous RL methods like Q-learning [40]. The actor network is to learn how to select action that results in higher reward, while the critic's role is to provide a baseline for actor's learning. Unlike deterministic approaches, PPO's actor network outputs the mean and variance of an action distribution rather than a specific action. The parameters of the actor network are updated by maximizing a surrogate objective function, aimed at increasing the probabilities of actions that maximize the expected cumulative rewards. Moreover, to maintain training stability and avoid excessive updates to network parameters during policy optimization, PPO employs a clipped surrogate objective function. This technique limits large updates, which can lead to instability. Additionally, the loss of the state value function, which is the difference of the actual and targeted state value, is trained by the critic to be minimized. Consequently, the objective function to be maximized at each training step is defined as

$$L_{t,\theta} = \hat{\mathbb{E}}_t \left[L_{t,\theta}^{CLIP} - \delta L_t^{VF} + \sigma S[\pi_\theta(s_t)] \right], \tag{3}$$

where $L_{t,\theta}^{CLIP} = \min\left(r_{t,\theta}\hat{A}_t,\ \text{clip}(r_{t,\theta}, 1-\epsilon, 1+\epsilon)\right)$ is the clipped surrogate function, $r_{t,\theta}$ is the probability ratio of the new policy to the old policy, $\hat{A}_t$ denotes the estimated advantage function at time t, δ and σ are the coefficients balancing the importance of the value function and the entropy, S is the entropy bonus, L_t^{VF} is the squared error loss of the value function estimated by the critic network, and ϵ is the hyper-parameter controlling the clipping range.

2.3 Multi-Agent Proximal Policy Optimization (MAPPO)

Algorithm 1: MAPPO algorithm

```
1  Randomly initialize the actor and critic networks.
2  for episode = 1, 2, ..., max_episode do
3      State initialization.
4      Buffer $D$ = [].
       /* Execution phase, illustrated in Fig. 2a and 3a. */
5      for $t$ = 1 to episode_length do
6          Each agent selects and executes action $a_t^k$.
7          Insert new data to $D$.
8      end
       /* Training phase, illustrated in Fig. 2b and 3b. */
9      Compute $\hat{R}$ and $\hat{A}$.
10     Split $D$ into small data chunks.
11     Select chunks randomly to form a mini-batch $B$.
12     Update $L(\phi)$ and $L(\theta)$ using $B$.
13 end
```

Algorithm 1 presents the MAPPO algorithm. The main operational logic behind MAPPO is the centralized training with decentralized execution framework, which means that the actors execute an action in their local area independently based on their local observation, while there is a shared critic that uses the global observation to guide the actors' update. In detail, at each execution step, the current local observation of the k-th agent, denoted as s_t^k, is the input to the actor. In the parameter-separated MAPPO, each agent has its own actor with the policy π_θ^k. As illustrated in Fig. 2a, each agent's actor receives its own state s_t^k, returns action a_t^k and the corresponding probability $\pi_\theta^k(a_t^k|s_t^k)$. On the other hand, for the parameter-sharing MAPPO, s_t^k is input to the shared actor π_θ, then this actor returns the

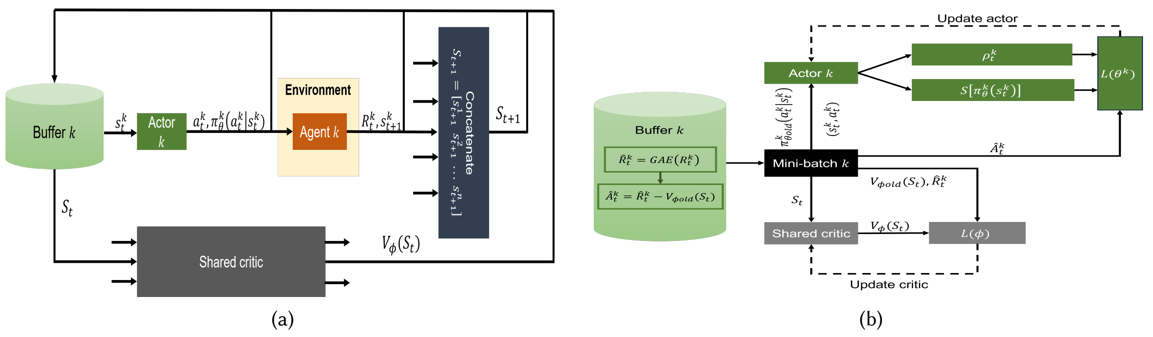


Fig. 2. Parameter-separated MAPPO's (a) execution flow, (b) training flow. Only one agent is illustrated in this figure for simplification. The operation of the other agents is the same as illustrated here.

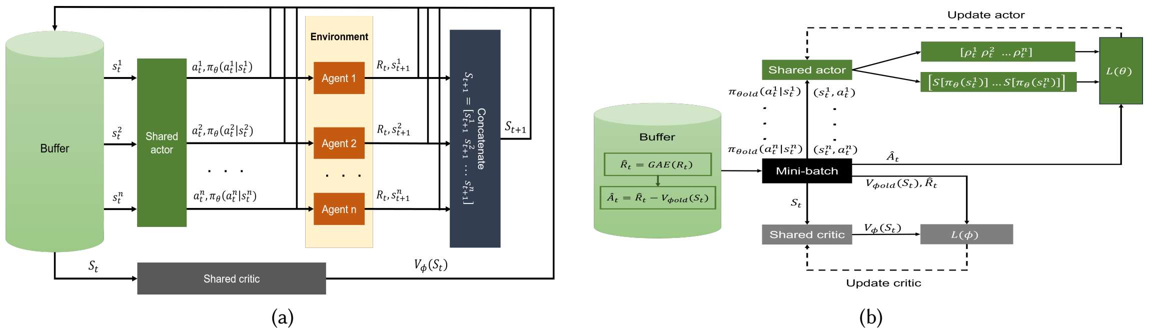

Buffer
s_t^1
s_t^2
s_t^n
Shared actor
$a_t^1, \pi_\theta(a_t^1|s_t^1)$
$a_t^2, \pi_\theta(a_t^2|s_t^2)$
$a_t^n, \pi_\theta(a_t^n|s_t^n)$
Environment
Agent 1
Agent 2
Agent n
R_t, s_{t+1}^1
R_t, s_{t+1}^2
R_t, s_{t+1}^n
$S_{t+1} = [s_{t+1}^1\ s_{t+1}^2 \dots s_{t+1}^n]$
Concatenate
S_{t+1}
S_t
Shared critic
$V_\phi(S_t)$
(a)
Update actor
Shared actor
$[\rho_t^1\ \rho_t^2 \dots \rho_t^n]$
$[S[\pi_\theta(s_t^1)] \dots S[\pi_\theta(s_t^n)]]$
$L(\theta)$
Buffer
$\hat{R}_t = GAE(R_t)$
$\hat{A}_t = \hat{R}_t - V_{\phi old}(S_t)$
$\pi_{\theta old}(a_t^1|s_t^1)$
(s_t^1, a_t^1)
$\pi_{\theta old}(a_t^n|s_t^n)$
(s_t^n, a_t^n)
Mini-batch
$\hat{A}_t$
$V_{\phi old}(S_t), \hat{R}_t$
S_t
Shared critic
$V_\phi(S_t)$
$L(\phi)$
Update critic
(b)

Fig. 3. Parameter-sharing MAPPO's (a) execution flow, (b) training flow.

corresponding action a_t^k and probability $\pi_\theta(a_t^k|s_t^k)$, as illustrated in Fig. 3a (hereafter, the parameter-sharing MAPPO's operation flow can be inferred from the parameter-separated MAPPO operation flow description by replacing π_θ^k with π_θ and θ^k with θ).

After choosing the action, the k-th agent executes a_t^k in the environment, receives reward R_t^k and observes new state s_{t+1}^k. Subsequently, global observation S_{t+1} is formed by combining the local ones together. As the action is selected and executed, the critic, which is shared amongst agents, estimates the value function V_ϕ based on the global observation S_t. By estimating the value based on S_t, the agent's policy is evaluated not only by its observation but also the information from other agents. At the end of the execution step, a_t^k, $\pi_\theta^k(a_t^k|s_t^k)$, R_t^k, s_{t+1}^k, S_{t+1}, and $V_\phi(S_t)$ are stored in the data buffer for later purposes (in the parameter-separated setting, each agent has its own buffer, while in the parameter-sharing setting, there is one buffer shared among the agents). After the episode is finished, in the buffer, the return value $\hat{R}_t^k$ is calculated by using the generalized advantage estimation (GAE) method [41], then the advantage value $\hat{A}_t^k = \hat{R}_t^k - V_\phi(S_t)$ is computed.

After executing the policy for an episode, the algorithm turns into the training phase, in which the actor and critic are updated with new data collected during the latest episode. Before training, the episode trajectory stored in the buffer is divided into smaller data chunks, which are compact, contiguous segments of the full episode. Then, these chunks are randomly sampled to form a mini-batch. For the actor training, firstly, the collected a_t^k, s_t^k and $\pi_{\theta old}^k(a_t^k|s_t^k)$ ($\pi_{\theta old}$ denotes the pre-trained policy) are passed through the actor to calculate probability ratio $\rho_t^k = \frac{\pi_\theta^k(a_t^k|s_t^k)}{\pi_{\theta old}^k(a_t^k|s_t^k)}$ and entropy $S[\pi_\theta^k(s_t^k)]$. After the whole mini-batch is processed, the actor loss is calculated by

$$L(\theta^k) = \frac{1}{Bn}\left[\sum_{t=1}^{B}\sum_{k=1}^{n} L_{t,\theta,k}^{CLIP} + \sigma \sum_{t=1}^{B}\sum_{k=1}^{n} S[\pi_\theta^k(s_t^k)]\right], \tag{4}$$

where $L_{t,\theta,k}^{CLIP} = \min\left[\rho_t^k \hat{A}_t^k,\ \text{clip}(\rho_t^k, 1-\epsilon, 1+\epsilon)\right]$, B is the mini-batch size, and σ is the entropy coefficient. Finally, the actor parameter θ^k is updated by using the Adam optimizer to minimize $L(\theta^k)$.

In the training phase of the parameter-separated MAPPO (illustrated in Fig. 2b), each actor processes the data from its own buffer only, that is, $n = 1$ in (4). On the other hand, in the parameter-sharing scheme, a_t^k, s_t^k and $\pi_{\theta old}(a_t^k|s_t^k)$ from all the agents are fed to one shared actor, as illustrated in Fig. 3b. Thus, for calculating the policy loss in the parameter-sharing scheme, n is equal to the number of agents, indicating the shared actor is trained by the mean loss calculated from all the agents.

Besides the policy training, the centralized critic is also trained, with the flow as follows: Firstly, from the mini-batch, the global observation S_t is sent to the critic to estimate the value $V_\phi(S_t)$. Then, $V_\phi(S_t)$ is used together with $\hat{R}_t^k$ and pre-collected $V_{\phi old}(S_t)$ to calculate the critic loss as

$$L(\phi) = \frac{1}{Bn}\sum_{t=1}^{B}\sum_{k=1}^{n} \max\left[(V_\phi(S_t) - \hat{R}_t^k)^2, (L_{\phi,t}^{CLIP})^2\right], \tag{5}$$

where $L_{\phi,t}^{CLIP} = \text{clip}(V_\phi(S_t), V_{\phi old}(S_t) - \epsilon, V_{\phi old}(S_t) + \epsilon)$ and n is the number of agents. Finally, the critic parameter ϕ is updated by minimizing $L(\phi)$.

From the above discussion, we can see that the implementation of the multi-agent framework, specifically the parameter-separated MAPPO, could enhance the training and execution efficiency of each agent by narrowing the state space and action space of each agent, thus reducing the amount of training data in the training phase and the amount of input in the execution phase. When partitioning the optimization task into several agents, the action and state dimensions of each agent are reduced by $O(n)$, where n is the number of agents. From the computational cost perspective, reducing an agent's action dimension by $O(n)$ would decrease its action computational cost by $O(a^n)$, where a is the computational cost to traverse over the search space of each variable. Similarly, when the state dimension of each agent is n times narrowed, each agent's computational cost to process the observation array would also exponentially decrease. Moreover, we deduce that the cooperation among agents can be achieved in MAPPO through these factors: (1) The shared critic, which uses the global observation as input in both its training and execution, thus reflecting the consequences of the collective actions of all agents; (2) Since the inclusion of the advantage function $\hat{A}_t^k$ into the policy loss function, the experience (i.e., rewards and observations) of the other agents can be included in the training of one agent. This cooperation can further be enhanced by applying the parameter-sharing scheme, where the actor loss is updated by the mean loss calculated from all the agents. However, the parameter-sharing setting could also degrade the learning efficiency, since putting too much complex information on one shared actor could result in inefficient modelling and learning. In subsequent sections, we will investigate whether the parameter-sharing MAPPO would enhance the learning efficiency or not, in the context of LDO design automation.

3 Sizing Automation Framework

3.1 Problem formulation

In analog IC sizing, the objective is to determine the geometric parameters of each transistor, such as width, length, and multiplier, as well as the nominal values of capacitors and resistors (whenever applicable). This process seeks to maximize (or minimize) one or more target performance metrics while ensuring that additional metrics meet specified constraint thresholds. Generally, analog IC sizing optimization problem can be mathematically formulated as follows:

$$\max_{x \in D^n} FoM(x), \tag{6}$$

where x represents the set of design parameters, D^n denotes the n-dimensional design space, and the Figure of Merit (FoM) is a function derived from the performance metrics intended for optimization. FoM can be *i)* a weighted sum, which aggregates all original objective functions with assigned weights, giving greater importance to the more significant objective function, or *ii)* created by multiplying or dividing the original objective functions [19].

In RL, agents are designed to maximize their reward functions. Similarly, in optimization problems, the objective is to maximize an objective function. This alignment allows the optimization objective to be naturally defined as the agents' reward when applying RL to optimization tasks.

3.2 Sizing Automation Framework Overview

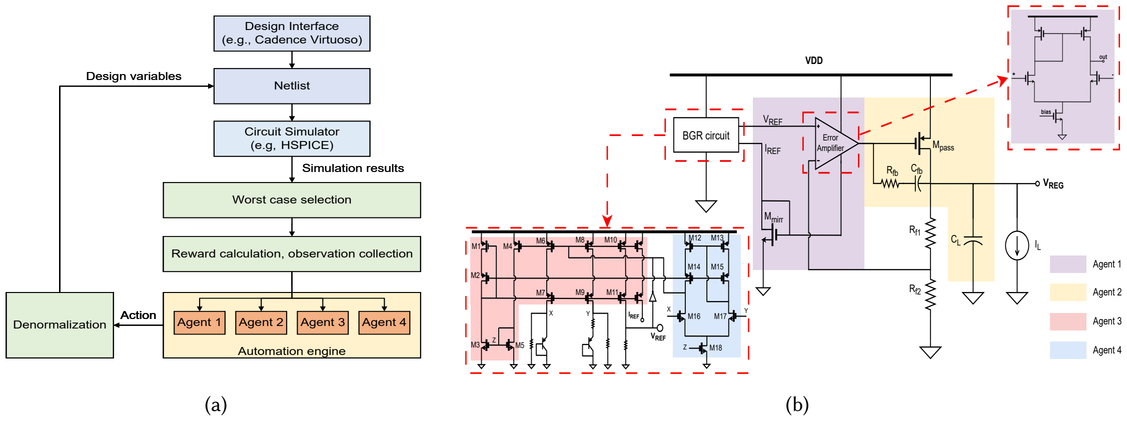


Fig. 4. (a) Implemented PPO/MAPPO-based sizing automation framework, (b) LDO module partition in MAPPO implementation.

Our proposed framework to implement PPO/MAPPO to tackle the LDO sizing automation problem is presented in Fig. 4a. The framework consists of two parts: the circuit simulator and the automation engine. In the circuit section, the LDO schematic with adjustable design parameters is created using a design interface (e.g., Cadence Virtuoso), which outputs a netlist file subsequently simulated with a circuit simulator (e.g., HSPICE). There are 28 design variables in total, which are the geometric parameters of the colored devices in Fig. 4b. For the transistors, designer variables are their width and length values, whose ranges are $[1\ \mu m, 100\ \mu m]$ and $[0.5\ \mu m, 3.0\ \mu m]$ respectively. Specifically for the transistors M_{pass} and M_{mirr}, the number of fingers is also treated as a design variable besides their width and length. For the passive devices, the number of multipliers of R_{fb}, C_{fb} and C_L are set as design variables. In the MAPPO implementation, the design variables are divided into four groups to be monitored by four agents, as illustrated in Fig. 4b. The variable division is determined by the functions of each group: *Agent 1* supervises the error amplifier; *Agent 2* monitors the devices related to the poles and zeros of the LDO's feedback loop; *Agent 3* oversees the devices that provide the current and bias voltage inside the BGR; and *Agent 4* looks after the BGR's op-amp, which plays a crucial role in creating its stable feedback loop. As a result of this module partition strategy, the number of variables under the supervision of *Agent 1* is 7, while that of *Agents 2*, *3*, and *4* is 6, 8, and 7, respectively. However, in the parameter-sharing MAPPO, one shared actor is used by all agents, thus requiring these agents to have a similar action dimension (i.e., the number of design variables). Therefore, to let the action dimension of all four agents be 8, that is, the number of variables under the management of *Agent 3*, the "dummy" variables, whose value is always unchanged, are introduced to *Agents 1*, *2*, and *4*. In detail, one "dummy" variable is added to *Agent 1*, two "dummy" variables are added to *Agent 2*, and one "dummy" variable is added to *Agent 4*.

The optimization engine uses one of three RL algorithms—PPO, parameter-sharing MAPPO, and parameter-separated MAPPO—all programmed in the Python programming language. Consistent hyperparameters, detailed in Table 2, are used across all algorithms to facilitate a fair comparison.

Table 2. Hyperparameter values employed among three RL algorithms

Hyperparameter	**Value**
Training step	10,240
Episode length	64, 128, 256
Number of mini-batch	4
Discount rate	0.9
GAE factor	0.8
PPO epoch	15
Actor learning rate	$3e-4$
Critic learning rate	$3e-4$
ϵ	0.2
δ	1
σ	0.01

The workflow of the framework is as follows: Firstly, the agents select actions. Due to the neural networks' activation function, these action values, denoted as $\hat{\boldsymbol{A}}$, fall in the interval [-1, 1]. To convert these action values back to the actual design values of the electrical components, the denormalization formula as follows is employed:

$$\boldsymbol{A} = \frac{1}{2}\left[\hat{\boldsymbol{A}} \odot (\boldsymbol{A}_{max} - \boldsymbol{A}_{min}) + (\boldsymbol{A}_{max} + \boldsymbol{A}_{min})\right], \tag{7}$$

where $\hat{\boldsymbol{A}}$ is the normalized action, $\boldsymbol{A}_{max}$ and $\boldsymbol{A}_{min}$ are the upper and lower bounds of the design variables, respectively. $[A_{min}, A_{max}] = [1, 100]$ for transistors' width, $[0.5, 3]$ for transistors' length, $[1, 20]$, $[1, 100]$, $[10, 300]$, $[10, 2000]$ and $[1, 20]$ for the multipliers of R_{fb}, C_{fb} and C_L, and the fingers of M_{pass} and M_{mirr}, respectively.

Once the denormalization is complete, every component of the action vector is mapped to its respective design variable within the netlist. Then, the netlist is used by the circuit simulator to simulate the circuit performance in three PVT corners - (25°C, TT), (-40°C, FF), and (125°C, SS) - respectively. Following the simulation, the worst-corner performance values are used to calculate the reward for the agents, and the transistors' operating point at the (25°C, TT) corner is sent back to the automation engine as observations.

3.3 Reward

In line with the reward formulation equation introduced in [42], we define the reward for each performance metric as

$$r_i = \begin{cases} \min\left(\frac{o_i - o_i^*}{o_i + o_i^*}, 0\right), & \text{if the spec requires } o_i \geq o_i^* \\ \min\left(\frac{o_i^* - o_i}{o_i + o_i^*}, 0\right), & \text{if the spec requires } o_i \leq o_i^* \end{cases}, \tag{8}$$

where o_i is the worst-corner simulation result of the metric and o_i^* is the target value of o_i. With this reward mechanism, if the specification is not met, r_i falls in the range of $(-1, 0)$. The closer the simulation value draws to its target, the higher the reward r_i gets. If the specification is satisfied, we will reach $r_i = 0$. This mechanism can encourage the agent to achieve performance metrics satisfying the predefined goals.

Moreover, some performance metrics need special attention because of their significance in certain applications. In LDO design, PSRR is an important metric since it measures how well the LDO rejects the ripple from the supply voltage. For LDO and BGR design, ensuring the stability of the inner feedback loop, which is represented by the phase margin (PM) metric, is also crucial. To prevent these critical metrics from falling to unsatisfactory levels, unique reward functions are designed for the PSRR specifications (Specs. 3, 4, 5, and 6 in Table 1) and the phase margin specifications (Specs. 7, 8, and 13 in Table 1), as listed as follows:

$$r_{PSRR} = \begin{cases} -1, & \text{if } PSRR \geq 0\ dB \\ \min\left(\frac{o_i^* - o_i}{o_i + o_i^*}, 0\right), & \text{if } PSRR < 0\ dB, \end{cases} \tag{9}$$

$$r_{PM} = \begin{cases} -1 + \min\left(\frac{o_i - o_i^*}{o_i + o_i^*}, 0\right), & \text{if } PM < 45^\circ \\ \min\left(\frac{o_i - o_i^*}{o_i + o_i^*}, 0\right), & \text{else.} \end{cases} \tag{10}$$

As discussed before, if a metric does not meet its specs, the normalized reward $\min\left(\frac{o_i^* - o_i}{o_i + o_i^*}, 0\right)$ (or $\min\left(\frac{o_i - o_i^*}{o_i + o_i^*}, 0\right)$) will have the value in the range of $(-1, 0)$. In (9), if the collected worst-corner PSRR value is greater than or equal to 0, which is considered as an "unacceptable" value, we will have $r_{PSRR} = -1$, which is smaller than $\frac{o_i^* - o_i}{o_i + o_i^*}$. That is to say, this would more actively encourage the agents to learn how to achieve a higher PSRR value. Similarly, for the phase margin metrics, the reward will be penalized by 1 if the collected phase margin value is less than 45°, as shown in (10), to diligently encourage the agents not to fall into this range.

The overall reward for each agent is determined by computing a weighted sum of the rewards associated with each performance metric, expressed as $R = \sum w_i r_i$, with an additional bonus of 10 if all $r_i = 0$. Thus, the higher R is, the closer the simulation results are maintained to the target, and R could have the highest value of 10. In the MAPPO implementation, since all the agents work together to make the whole circuit achieve the predefined specification, we set the same reward R for all agents, for both the parameter-sharing and parameter-separated implementations. R is also set as the reward for the agent in the single-agent PPO implementation.

3.4 Observation

In this study, the observation is to capture the operating point of colored transistors in Fig. 4b. The operating point of each transistor is represented by a vector called S_i, containing 7 critical electrical characteristics: drain-source current I_{DS}, transconductance g_m, drain-source conductance g_{DS}, drain-source voltage V_{DS}, threshold voltage V_{th}, and saturation voltage V_{dsat}. These parameters collectively define the operation status of each transistor and serve as inputs for the training and decision-making process of the RL agents. In MAPPO implementation, the local observation of each agent is the operating point of all the transistors upon which it executes an action. In parameter-sharing MAPPO, since the actor is shared amongst agents, these agents must have an identical observation dimension. In this study, we set the size of the observation array of each agent in MAPPO as 6×7 (6 is the number of transistors under the supervision of *Agent 1*), and the observation dimension in the single-agent PPO implementation is 24×7. Each row of the observation array observes the operating point of a transistor within the sub-block assigned to the corresponding agent. However, not all agents supervise exactly 6 transistors, which does not align with the observation dimension of 6×7. To overcome this difficulty, the following technique is applied: For *Agent 2*, the operation vector of transistor M_{pass} is replicated across six rows of the observation matrix. For any agents that supervise more than 6 transistors (e.g., *Agent 3* or *4*), only one representative transistor from the group staying in the same Cascode stage is included in the observation array, since transistors within the same Cascode stage typically have identical operating points. In detail, for the block under the supervision of *Agent 3*, operating points of *M1*, *M2*, *M3*, *M4*, *M6*, *M7* (in Fig. 4b) are determined in the observation array; and for *Agent 4*, operating points of *M12*, *M13*, *M14*, *M16*, *M17*, *M18* are determined in its observation. The global observation is the concatenation of local observations.

To improve the learning capacity of the RL algorithm and ensure consistent scaling, the observations are normalized as follows:

$$\hat{S}_i = \frac{S_i - \mu}{\sigma}, \tag{11}$$

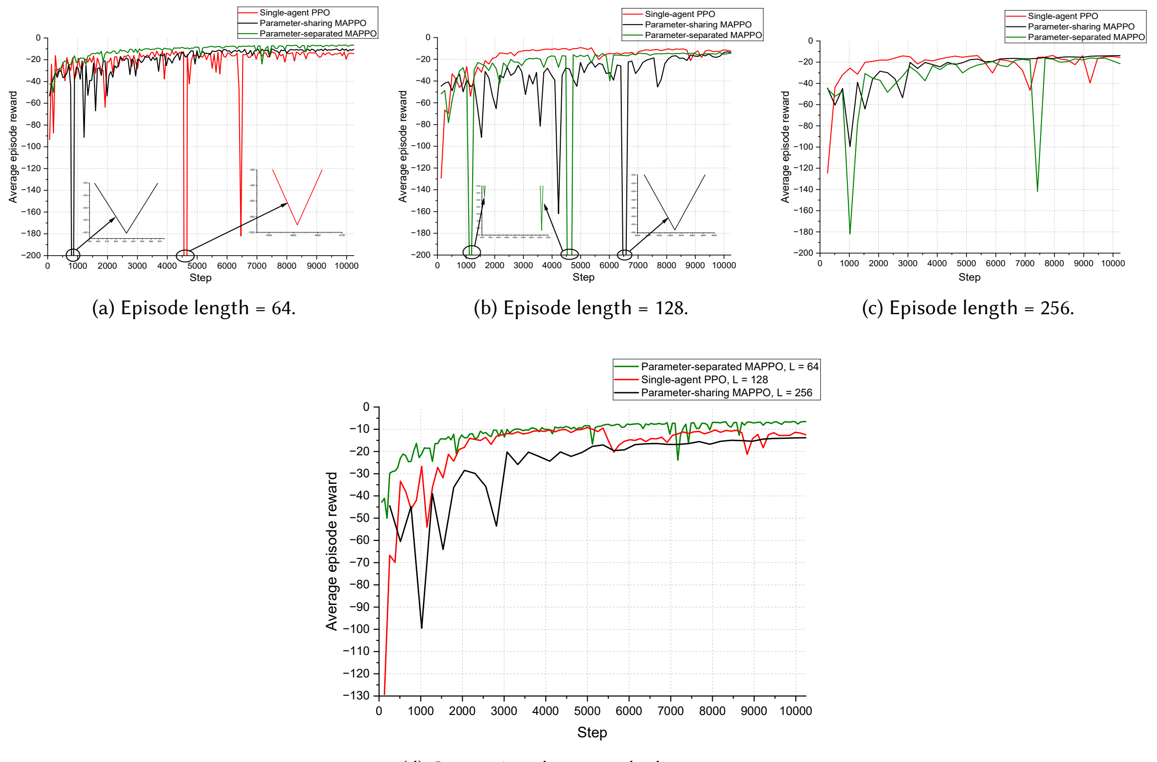
where μ and σ denote the mean and variance, respectively, of the transistor's DC operating point collected from 100 random simulations.

Following normalization, each $\hat{S}_i$ is clipped within the range of $[-5, 5]$ to mitigate the impact of outliers and ensure stable learning. This normalization and clipping process ensures that the observation values remain within a tractable range for the neural network, preventing extreme values from disproportionately affecting the training process and fostering a more robust learning environment.

4 Experimental Result and Discussion

Fig. 5 depicts the learning curve of the single-agent PPO, the parameter-sharing MAPPO, and the parameter-separated MAPPO over 10,240 steps. To investigate how different hyperparameter settings could affect the algorithms' performance, we have run the three algorithms with episode lengths of 64, 128, and 256, whose results are presented in Figs. 5a, 5b, and 5c, respectively.

At the episode length setting of 64, as presented in Fig. 5a, generally, the learning curves of all three algorithms grow gradually and fluctuate increasingly less. The parameter-sharing and parameter-separated MAPPO demonstrate stable learning curves after 5,000 steps, showing that these algorithms have learnt stable policies. For the single-agent PPO, the mean reward values measured at steps 4,608 and 6,464 drop significantly, but then quickly recover. This behavior might be due to the RL algorithm exploring an action result in a malfunctional circuit, receiving a bad reward, and then recovering to the old policy. In the final episode, the parameter-separated MAPPO gains the highest average episode reward (-6.57) compared to the other algorithms. When the episode length is set to 128, shown in Fig. 5b, generally, the learning curve of the parameter-separated MAPPO shows a stable policy after 6,144 steps, while the parameter-sharing MAPPO and the single-agent PPO require more than 9,000 steps to be stable. The three methods achieve similar average episode rewards at the final episode (-12.50 for single-agent PPO, -13.76 for parameter-sharing MAPPO, and -14.40 for parameter-separated MAPPO). In the case of 256 as the episode length, as shown in Fig. 5c, generally the learning curve of the parameter-sharing MAPPO is less fluctuating than that of the episode length with 128. However, the



(a) Episode length = 64.

(b) Episode length = 128.

(c) Episode length = 256.

(d) Comparison between the best cases.

Fig. 5. Average episode reward vs. the number of learning steps for three PPO algorithms at different episode length L.

average episode reward of the other algorithms (i.e., single-agent PPO and parameter-separated MAPPO) shows more fluctuating learning, indicating less stable learning curves.

From Fig. 5, we can see that with the same number of execution steps, the learning curves seem to fluctuate more when the episode length is larger. This result is actually explainable, as analyzed below: if the episode is longer and the total execution steps remains the same, the algorithm needs to process more data in each training phase. Thus, there would be fewer training steps. Fig. 5d compares the best curves picked from Figures 5a, 5b, and 5c, demonstrating that the parameter-separated MAPPO reaches the highest average episode reward in the end. The higher the average episode reward curve reaches, the more close-to-target design points the algorithm can propose.

To investigate the exploration ability of each algorithm, Fig. 6 presents the maximum reward that each algorithm has recorded during training the three PPO algorithms, with different episode length settings for each algorithm. Besides that, Fig. 6 also includes the result obtained from running the particle swarm optimization (PSO) algorithm [43] with 160 updating steps, 64 particles for each step (thus allowing PSO to collect 64 different data points for each learning step and run 10,240 data points for the whole training process, analogous to setting the RL algorithms with 10,240 training steps holding episode length of 64). Generally, all of the maximum reached reward values are smaller than 0, showing that none of the algorithms can find a design point that satisfies all the constraints. With the same number of simulations and similar runtime, all of the PPO-based algorithms record the maximum reward higher than that of

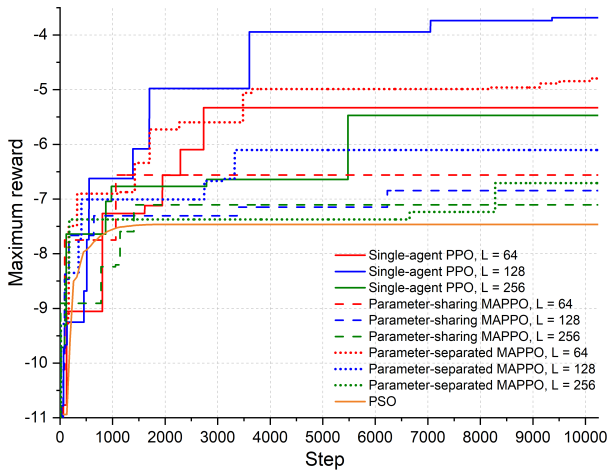


Fig. 6. Maximum reward comparison.

Table 3. LDO worst-corner performance, resulting from the three PPO algorithms, the PSO algorithm, and the commercial tool. The corresponding sizing is presented in Table 5 in the Appendix.

No.	Metric	Single-agent PPO	Parameter-sharing MAPPO	Parameter-separated MAPPO	PSO	Cadence ADE GXL Global Optimization	Constraint	Unit
1	V_d	234.00	206.00	220.00	222.00	203.10	≤ 220	mV
2	V_L	29.11	28.34	10.41	31.40	20.43	≤ 36	mV
3	$PSRR_{1kHz,HL}$	-30.83	-31.80	-30.98	-27.40	-39.91	≤ -30	dB
4	$PSRR_{1kHz,LL}$	-31.28	-32.40	-32.45	-33.26	-41.43	≤ -30	dB
5	$PSRR_{1MHz,HL}$	0.10	-0.04	-0.56	0.02	0.02	≤ -20	dB
6	$PSRR_{1MHz,LL}$	-1.53	-2.41	-0.61	0.87	-0.75	≤ -20	dB
7	PM_{HL}	83.39	89.20	93.94	89.44	89.78	≥ 60	$^\circ$
8	PM_{LL}	63.12	47.12	71.49	56.33	71.41	≥ 60	$^\circ$
9	I_q	11.62	34.93	21.96	24.72	22.24	≤ 200	μA
10	TC	17.60	55.87	29.36	33.83	25.01	≤ 10	ppm/$^\circ$C
11	V_{REF}	0.890	0.903	0.901	0.904	0.898	$= 0.9 \pm 0.02$	V
12	LG_{BGR}	42.30	40.09	40.56	42.30	53.28	≥ 40	dB
13	PM_{BGR}	66.59	59.24	70.28	68.31	37.08	≥ 60	$^\circ$
	Runtime	3 days 11 hours	3 days 11 hours	3 days 14 hours	3 days 7 hours	23 hours 17 minutes		
	No. of simulations	10,240	10,240	10,240	10,240	3,150		
	No. of successful spec	9	9	10	7	9		

the PSO algorithm, indicating that the exploration capability of PPO-based algorithms outperforms that of the PSO algorithm. The more efficient the exploration of the algorithm is, the higher ability it features to find a good design point. Comparing PPO-based algorithms and different episode length settings, the single-agent PPO with the episode length setting of 128 shows the most efficient exploration ability. It needs just 1,702 steps to find a point leading to the reward of -4.97, 3,603 steps to find a point resulting in the reward of -3.94, and its maximum reward recorded after 10,240 steps is -3.68, higher than all other test cases. The parameter-separated MAPPO with episode length of 64 earns the second-highest maximum reward. Within the parameter-sharing MAPPO setting, the episode length of 64 can find the highest reward. With the exception of the single-agent PPO with the episode length of 128 and the parameter-separated MAPPO with episode length of 64, all other test cases conclude the training process with the recorded maximum rewards lower than -5.00.

As discussed in Section 2.3, the implementation of parameter-sharing MAPPO could result in better or worse performance compared to parameter-separated MAPPO and single-agent PPO. The sharing mechanism can offer more comprehensive information to agents, which may boost their learning performance. On the other hand, in an environment where the efficiency of an agent is not significantly affected by other agents, the sharing mechanism could degrade the learning performance, as it allows the agent to process irrelevant information. In this LDO design automation study, the results obtained from the parameter-separated MAPPO and single-agent PPO outperform those of the parameter-sharing MAPPO, in terms of both the final average episode reward value and the maximum reward reached. However, looking at a bigger circuit where the relevance among its sub-blocks is tighter, the cooperation among agents in the parameter-sharing MAPPO could be helpful in improving the circuit performance, as the sharing mechanism in parameter-sharing MAPPO can allow agents to balance the performance among all the sub-blocks, not just focusing on their own.

Table 3 presents the performance of the LDO circuit resulting from running three PPO-based (single-agent PPO, parameter-separated MAPPO, parameter-sharing MAPPO) sizing automation algorithms, the PSO algorithm, and the Cadence Virtuoso ADE GXL Global Optimization tool (with the worst-corner performance as listed here). Using its developed algorithm, the Cadence commercial tool runs 3,150 simulations in 23 hours, with the design space and specifications similar to those of the PPO-based automation engine. From Table 3, it can be seen that neither the developed algorithms nor the commercial Cadence tool can satisfy all specifications. V_d is not satisfied by the single-agent PPO and the PSO, while V_L values resulting from all the developed algorithms and the Cadence tool satisfy the requirement. For the $PSRR_{1MHz,HL}$, PM_{HL}, and TC specifications, none of the methods above can meet the constraints. Regarding the PM_{BGR} metric, which requires a value larger than 60°: the single-agent PPO and the parameter-separated MAPPO meet the requirement, while the Cadence tool does not, and the parameter-sharing MAPPO reaches a value (59.24°) that is very close to the threshold (60°). The last row of Table 3 shows the number of satisfied specifications for each sizing automation approach. The result proposed by the parameter-separated MAPPO satisfies 10/13 constraints, higher than that of the single-agent PPO (9/13), the parameter-sharing MAPPO (9/13), the Cadence optimization tool (9/13), and the PSO (7/13). It is worth mentioning that the listed runtime (i.e., in the range of 3 days and 7-14 hours) of the PPO-based and PSO methods in Table 3 must not be interpreted as a proof of their slowness compared to the Cadence optimization tool. As the main focus of this work is to investigate the optimization effectiveness and convergence capability of various methods, in our experiments we have deliberately defined the fixed number of simulations (i.e., 10,240) in the running of the PPO-based and PSO methods. But for the Cadence optimization tool, we could not control its total simulation number and termination condition in the experiment. In this regard, looking at the convergence flow as depicted in Fig. 6, one can observe that the PPO-based methods typically complete the major performance boosting among the first 3,000-3,600 simulations, which is around the level similar to the Cadence optimization tool (showing 3,150 simulations completed as read from Table 3).

The outcome resulting from our proposed PPO-based automation framework, especially the parameter-separated MAPPO one, is promising. Within a similar running time, it outperforms the PSO in exploring the design space to find a better reward, resulting in the circuits having better performance (satisfying more constraints). Moreover, the result from parameter-separated MAPPO is not worse than that of the Cadence tool. The parameter-separated MAPPO achieves a success rate of 10/13, higher than that of the Cadence tool (9/13). Comparing the parameter-sharing MAPPO and the Cadence tool, despite reaching the same number of successful specifications (9/13), the parameter-sharing MAPPO's phase margin metric is closer to the target than that of the Cadence tool (59.26° compared to 37.08°). Given these marginal improvements in design performance, the proposed MAPPO framework shows potential to serve as an alternative choice for the individuals who experience difficulty in accessing the commercial tools. On the other hand, despite deriving better results, the PPO-based algorithms need to consume more runtime (i.e., maximally 8% slower than PSO as listed in Table 3). The PPO-based algorithms employ the neural network in their actors and critic, while the PSO method follows a stochastic population-based optimization flow and the Cadence tool is based on a parallel simulated annealing algorithm [44]. The neural network requires data to model the behavior of the learned function, while the conventional methods such as simulated annealing use a strategy to sample over the design domain to propose an approximate solution [45]; therefore, the neural network requires more time to collect data but is more accurate. In a simple design automation task, users may prefer using the heuristic-based algorithms like PSO or simulated annealing since they require less runtime; but if looking at more complex design tasks, the machine-learning-based algorithms like PPO or MAPPO would be more preferred because their results can outperform the heuristic-based algorithms in terms of accuracy by virtue of the prowess from the neural network modelling.

Besides the marginal improvement in design results, another nontrivial advantage of our framework compared to the Cadence ADE Global Optimization tool can be the transparency and the controllability. For the Cadence commercial tool, neither the detailed algorithm, the hyperparameter selection, nor how the automation engine selects the number of simulations required (i.e., 3,150 as stated before) is clear to the user, making it hard for the user to fine-tune the hyperparameters or run more simulations to more thoroughly discover the design space. In contrast, for our developed algorithms, the number of learning steps and the implemented hyperparameters are clear and can be easily changed. Moreover, with our developed PPO-based algorithms, users can determine whether more simulations are necessary when looking at the average episode reward versus number of training steps plot: if no significant increase or fluctuation of the learning curve is seen after a certain time, we can conclude that the algorithm has converged and further simulation could not contribute to finding a better outcome.

Table 4 provides a comparative analysis of our research with other published works. Our proposed scheme demonstrates the capability to efficiently manage a larger number of specifications (13) as opposed to the two earlier studies, which manage only 6 specifications. Furthermore, our approach is capable of handling 28 variables, in contrast to the conventional methodologies employed in prior studies.

Table 4. Comparison with the state-of-the-art works

	DAC'20[32]	ICCAD'23[34]	**This work**
Algorithm	DDPG	DDPG	PPO MAPPO
Circuit	LDO	LDO	LDO
Number of specifications	6	6	13
Number of variables	N/A	13	28

5 Conclusion

We presented the implementation of three PPO-based algorithms - the single-agent PPO, the parameter-separated MAPPO, and the parameter-sharing MAPPO - in addressing the LDO circuit sizing automation problem. The experimental results showed that the PPO implementations outperformed the classical PSO algorithm in handling the LDO sizing automation task in terms of exploration ability and number of satisfied specifications in the end. Comparing the various PPO-based algorithms with different episode length settings, we identified that for the single-agent PPO, an episode length of 128 resulted in the highest average episode reward and the least fluctuating learning curve, and for the parameter-separated and parameter-sharing MAPPO, these were achieved by setting the episode length to 64 and 256, respectively. The experimental result also showed that the parameter-separated MAPPO with the episode length of 64 not only achieved the highest average episode reward and gained the fastest convergence rate among all algorithms and settings (indicators of the most effective learning process), but also showed the most efficient exploration capability similar to the single-agent PPO with the episode length of 128. Compared to the design result obtained by using the Cadence ADE Optimization tool, our proposed MAPPO design automation framework demonstrates a marginal improvement, showing its promise to be an in-house low-cost design automation tool available beyond the commercial ones.

Acknowledgements

This work was supported in part by the Canada Excellence Research Chair (CERC) Program CERC-2022-00109, in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), Canada Foundation for Innovation (CFI), Provincial Government of Newfoundland and Labrador, and Memorial University of Newfoundland.

This paper has been accepted in part for presentation at the IEEE International Symposium on Circuits and Systems (IEEE ISCAS 2025), London, UK, May 2025 [46].

References

[1] P. Manikandan and B. Bindu, “A Review on Frequency Compensation and Transient Enhancement Schemes of Flipped Voltage Follower LDO Regulators for SoC Applications,” *IEEE Access*, Jan. 2025.

[2] N. Zachos, V. Gogolou, and T. Noulis, “A Fully Integrated 1.8 V Low-Power LDO Regulator with Dynamic Transient Control for SoC Applications,” *Electronics*, vol. 13, no. 23, p. 4734, Nov. 2024.

[3] M. Reza, N. Alam, and S. J. Gaggatur, “A 0-24mA, 1.2V/1.8V Dual Mode Low Dropout Regulator Design for Efficient Power Management in Battery-Powered Systems,” in *Proc. 2024 Int. Symp. VLSI Des. Test (VDAT)*, Sep. 2024, pp. 1–6.

[4] H. M. Nguyen, L. D. Pham, and T. Hoang, “A novel Li-ion battery charger using multi-mode LDO configuration based on 350 nm HV-CMOS,” *Analog Integr. Circuits Signal Process*, vol. 88, pp. 505–516, Jun. 2016.

[5] X. Dong and L. Zhang, “PV-Aware Analog Sizing for Robust Analog Layout Retargeting with Optical Proximity Correction,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 23, no. 6, pp. 71:1–19, Nov. 2018.

[6] Y. Tsunoda, C. Tsuchiya, Y. Segawa, H. Sawaya, M. Hasegawa, S. Ishigaki, and K. Ishibashi, “A Small-Size Energy-Harvesting Electric Power Sensor for Implementing Existing Electrical Appliances Into HEMS,” *IEEE J. Solid-State Circuits*, vol. 16, no. 2, pp. 457–463, Jan. 2016.

[7] H. Park, W. Jung, M. Kim, and H.-M. Lee, “A Wide-Load-Range and High-Slew Capacitor-Less NMOS LDO With Adaptive-Gain Nested Miller Compensation and Pre-Emphasis Inverse Biasing,” *IEEE J. Solid-State Circuits*, vol. 58, no. 10, pp. 2696–2708, Oct. 2023.

[8] D. Li, Y. Zhou, U. R. Tida, Z. Liu, and X. Zou, “An Energy-Efficient LDO for Biomedical Implantable Devices,” in *Proc. IEEE Int. Conf. Circuits Syst. (ICCS)*, Oct. 2021, pp. 242–246.

[9] J. Yuan, W. Fan, W. Sun, and Z. Fang, “A CMOS-Integrated Capacitor-Less N-LDO Featuring 1.2-1.8V Output Voltage Range and Low FOM for Edge Biomedical Device Applications,” in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, 2024, pp. 1–5.

[10] J. S. Kim, K. Javed, and J. Roh, “Design of a Low-Power and Area-Efficient LDO Regulator Using a Negative-R-Assisted Technique,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 10, pp. 3892–3896, Jun. 2023.

[11] M. Sekyere and D. Chen, “Low Power, Fully-Integrated Flipped Voltage Follower LDO Using Off-State Non-Linear Circuits for Enhanced Transient Performance,” in *Proc. 2024 IEEE Int. Mid. Symp. Circuits Syst. (MWSCAS)*, Aug. 2024, pp. 9347–9351.

[12] Y. Zhang, J. Cai, J. Chen, and Y. Yin, "A 640 nA IQ Output-Capacitor-Less Low Dropout (LDO) Regulator with Sub-Threshold Slew-Rate Enhancement for Narrow Band Internet of Things (NB-IoT) Applications," *Micromachines*, vol. 15, no. 8, p. 1019, Aug. 2024.

[13] H. Ameziane, K. Zared, H. Akhmal, and H. Qjidaa, "CMOS LDO Regulator with Improved Performance for lot and Wearable Devices," in *Proc. Int. Conf. Elect. Syst. and Automat.*, vol. 2. Springer Nature, 2024, p. 297.

[14] A. Privat, P. W. Davis, H. J. Barnaby, and P. C. Adell, "Total Dose Effects on Negative and Positive Low-Dropout Linear Regulators," *IEEE Trans. Nucl. Sci.*, vol. 67, no. 7, pp. 1332–1338, Feb. 2020.

[15] Z. Zhao and L. Zhang, "Analog Integrated Circuit Topology Synthesis With Deep Reinforcement Learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 12, pp. 5138–5151, Feb. 2022.

[16] T. Liao and L. Zhang, "Efficient Parasitic-aware gm/ID-based Hybrid Sizing Methodology for Analog and RF Integrated Circuits," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 26, no. 2, pp. 1–31, Oct. 2020.

[17] M. Sadrafshari, O. Dobre, and L. Zhang, "Reinforcement-learning-based foggy-aware optimal placement method for analog and mixedsignal circuits," in *Proc. IEEE Int. Symp. Circuits Sys. (ISCAS)*, May 2024, pp. 1–5.

[18] M. Torabi and L. Zhang, "A fast hierarchical adaptive analog routing algorithm based on integer linear programming," *ACM Trans. Des. Automat. Electron. Syst. (TODAES)*, vol. 22, no. 3, pp. 1–23, May 2017.

[19] T. Q. Nguyen, T. Hoang, L. Zhang, O. A. Dobre, and T. Q. Duong, "A Survey on Smart Optimisation Techniques for 6G-oriented Integrated Circuits Design," *Mobile Netw. Appl.*, vol. 28, p. 2227–2244, May 2024.

[20] H. Omran, M. H. Amer, and A. M. Mansour, "Systematic Design of Bandgap Voltage Reference Using Precomputed Lookup Tables," *IEEE Access*, vol. 7, pp. 100 131–100 142, Jul. 2019.

[21] A. A. Youssef, B. Murmann, and H. Omran, "Analog IC Design Using Precomputed Lookup Tables: Challenges and Solutions," *IEEE Access*, vol. 8, pp. 134 640–134 652, Jul. 2020.

[22] Y. R. AwadAllah, K. Y. Yasseen, and H. Omran, "Design Automation of CMOS Amplifiers Using Precomputed Lookup Tables: A Comparative Study of optimization Algorithms," in *Proc. Int. Telecomm. Conf. (ITC-Egypt)*, Alexandria, Egypt, Jul. 2022, pp. 1–6.

[23] K. Elmeligy and H. Omran, "Fast design space exploration and multi-objective optimization of wide-band noise-canceling LNAs," *Electronics*, vol. 11, no. 5, p. 816, Mar. 2022.

[24] T. Hoang, T. Q. Nguyen, and H. Phan-Tran-Minh, "Novel GA-OCEAN Framework for Automatically Designing the Charge-Pump Circuit," *IEEJ Trans. Electr. Electron. Eng.*, vol. 19, no. 10, pp. 1711–1719, Oct. 2024.

[25] T. Hoang, T. N. Quoc, L. Zhang, and T. Q. Duong, "Novel Methods for Improved Particle Swarm Optimization in Designing the Bandgap Reference Circuit," *IEEE Access*, vol. 11, pp. 139 964–139 978, Dec. 2023.

[26] S. K. Dash, B. P. De, B. Appasani, and N. Rout, "Design of Area-Optimized Low-power Voltage Reference Circuits using a Modified Particle Swarm Optimization," *J. Electron. Mater.*, pp. 1–13, Nov. 2024.

[27] C. Chen, H. Wang, X. Song, F. Liang, K. Wu, and T. Tao, "High-Dimensional Bayesian Optimization for Analog Integrated Circuit Sizing Based on Dropout and gm/ID Methodology," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 11, pp. 4808–4820, Jan. 2022.

[28] S. Zhang, W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Bayesian optimization approach for analog circuit synthesis using neural network," in *Proc. Design, Automat. Test Europe Conf. Exhibit*, Florence, Italy, Mar. 2019, pp. 1463–1468.

[29] H. Wang, J. Yang, H.-S. Lee, and S. Han, "Learning to Design Circuits," *arXiv preprint arXiv:1812.02734*, Dec. 2020.

[30] Z. Li and A. Chan Carusone, "An Open-Source AMS Circuit Optimization Framework Based on Reinforcement Learning—From Specifications to Layouts," *IEEE Access*, vol. 12, pp. 150 032–150 045, Oct. 2024.

[31] Z. Zhao and L. Zhang, "Deep Reinforcement Learning for Analog Circuit Sizing," in *Proc. IEEE Int. Symp. on Circuits and Syst. (ISCAS)*, Seville, Spain, Oct. 2020, pp. 1–5.

[32] 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. ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jul. 2020, pp. 1–6.

[33] K. Settaluri, Z. Liu, R. Khurana, A. Mirhaj, R. Jain, and B. Nikolic, "Automated Design of Analog Circuits Using Reinforcement Learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 9, pp. 2794–2807, Oct. 2022.

[34] Z. Li and A. C. Carusone, "Design and Optimization of Low-Dropout Voltage Regulator Using Relational Graph Neural Network and Reinforcement Learning in Open-Source SKY130 Process," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, San Francisco, CA, USA, Oct. 2023, pp. 01–09.

[35] J. Pérez-Bailón, B. Calvo, and N. Medrano, "A Fully-Integrated 180 nm CMOS 1.2 V Low-Dropout Regulator for Low-Power Portable Applications," *Electronics*, vol. 10, no. 17, p. 2108, Aug. 2021.

[36] A. A. Khan and R. K. Palani, "Analysis and Design of Low Noise Voltage Regulator With Integrated Single BJT Bandgap Reference Upto 10 mA Loads," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 71, no. 6, pp. 2966–2970, Jan. 2024.

[37] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games," in *Proc. Conf. Neural Inf. Process. Syst.*, New Orleans, LA, USA, Nov. 2022.

[38] J. Zhang, J. Bao, Z. Huang, X. Zeng, and Y. Lu, "Automated Design of Complex Analog Circuits with Multiagent based Reinforcement Learning," in *Proc. ACM/IEEE. Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2023, pp. 1–6.

[39] J. Bao, J. Zhang, Z. Huang, Z. Bi, X. Feng, X. Zeng, and Y. Lu, "Multiagent Based Reinforcement Learning (MA-RL): An Automated Designer for Complex Analog Circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 43, no. 12, pp. 4398–4411, 2024.

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

[41] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-dimensional continuous control using generalized advantage estimation,” *arXiv preprint arXiv:1506.02438*, Jun. 2018.

[42] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolic, “AutoCkt: Deep Reinforcement Learning of Analog Circuit Designs,” in *Proc. Design, Automat. Test Europe Conf. Exhibit. (DATE)*, Grenoble, France, Mar. 2020, pp. 490–495.

[43] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proc. Int. Con. Neural Netw. (ICNN)*, vol. 4, Perth, WA, Australia, Dec. 1995, pp. 1942–1948.

[44] Cadence Design Systems, Inc., “Virtuoso Analog Design Environment XL User Guide,” 2014. [Online]. Available: https://picture.iczhiku.com/resource/eetop/syIfptILiLPyrvCB.pdf

[45] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization: Overview and conceptual comparison,” *ACM Comput. Surveys*, vol. 35, no. 3, pp. 268–308, Sep. 2003.

[46] T. Q. Nguyen, L. Zhang, O. A. Dobre, T. Hoang, and T. Q. Duong, “Multi-Agent Proximal Policy Optimization Applications in Low-Dropout Regulator Design,” in *Proc. IEEE Int. Symp. on Circuits and Syst. (ISCAS)*, London, UK, May 2025, pp. 1–5.

APPENDIX

Table 5. Sizing point resulting from the three PPO-based algorithms, the PSO algorithm, and the Cadence tool. The width (W) and length (L) values are in μm. Implementation of design variables on circuit netlist is illustrated in Fig. 7.

Variable	**Single-agent PPO**	**Parameter-sharing MAPPO**	**Parameter-separated MAPPO**	**PSO**	**Cadence ADE GXL Global Optimization**
W_{M1}	13.7	1.0	1.0	1.0	2.0
L_{M1}	3.0	0.5	0.5	0.8	2.5
W_{M3}	97.2	100.0	100.0	70.4	18.0
L_{M3}	3.0	2.4	1.57	0.5	0.5
W_{M5}	1.0	92.0	60.4	43.5	63.0
L_{M5}	2.6	2.4	3.0	1.8	2.0
W_{Mpass}	98.8	100.0	23.75	39.2	51.0
L_{Mpass}	0.5	0.5	0.5	2.7	1.0
M_{Mpass}	2000	1398	563	695	353
M_{CL}	196	292	300	1	222
M_{Rfb}	20	1	1	16	5
M_{Cfb}	100	100	100	73	89
$W_{M1,BGR}$	3.0	7.6	10.0	5.5	10.0
$L_{M1,BGR}$	3.0	3.0	0.5	3.0	2.0
$W_{M2,BGR}$	80.0	61.0	34.7	67.6	38.0
$L_{M2,BGR}$	2.3	2.1	0.5	2.4	1.5
$W_{M3,BGR}$	30.0	24.1	30.0	29.6	21.0
$L_{M3,BGR}$	2.8	0.5	2.2	2.2	3.0
$W_{M4,BGR}$	80.0	64.7	80.0	30.0	54.0
$L_{M4,BGR}$	3.0	2.9	3.0	3.0	2.0
$W_{M12,BGR}$	30.0	30.0	3.4	8.0	13.0
$L_{M12,BGR}$	0.9	1.2	0.5	0.9	3.0
$W_{M14,BGR}$	74.6	38.9	51.7	77.2	49.0
$L_{M14,BGR}$	0.7	3.0	0.8	2.3	1.5
$W_{M16,BGR}$	53.9	15.1	80.0	50.9	44.0
$L_{M16,BGR}$	0.8	1.8	0.5	1.1	2.5
$W_{M18,BGR}$	31.2	56.1	26.5	32.1	26.0
M_{Mmirr}	2	20	1	8	4

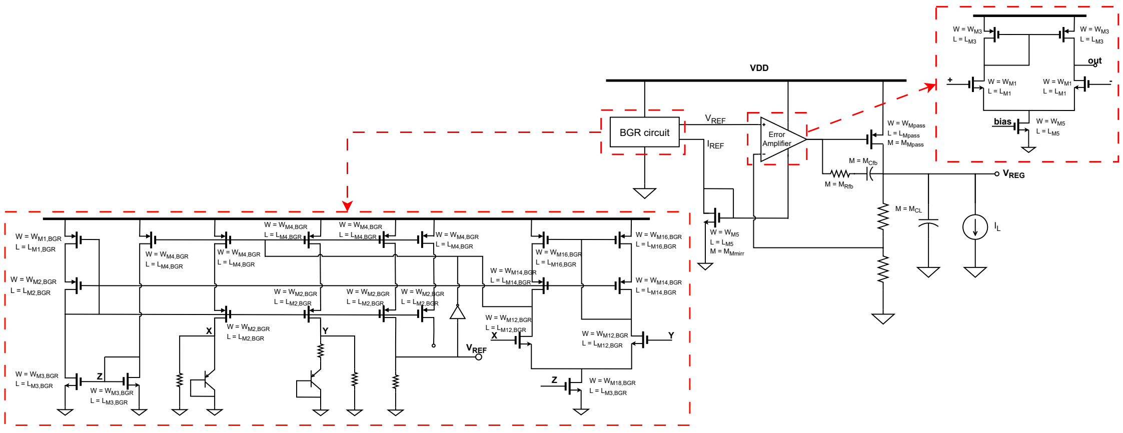

VDD
BGR circuit
V_{REF}
I_{REF}
Error Amplifier
$W = W_{Mpass}$ $L = L_{Mpass}$ $M = M_{Mpass}$
$M = M_{Cfb}$
$M = M_{Rfb}$
V_{REG}
$M = M_{CL}$
I_L
$W = W_{M5}$ $L = L_{M5}$ $M = M_{Mmirr}$
$W = W_{M3}$ $L = L_{M3}$
$W = W_{M1}$ $L = L_{M1}$
out
bias
$W = W_{M5}$ $L = L_{M5}$
$W = W_{M1,BGR}$ $L = L_{M1,BGR}$
$W = W_{M2,BGR}$ $L = L_{M2,BGR}$
$W = W_{M3,BGR}$ $L = L_{M3,BGR}$
$W = W_{M4,BGR}$ $L = L_{M4,BGR}$
$W = W_{M12,BGR}$ $L = L_{M12,BGR}$
$W = W_{M14,BGR}$ $L = L_{M14,BGR}$
$W = W_{M16,BGR}$ $L = L_{M16,BGR}$
$W = W_{M18,BGR}$ $L = L_{M3,BGR}$
X
Y
Z
V_{REF}

Fig. 7. Implementation of design variables on the circuit netlist.