EVDMARL: Efficient Value Decomposition-based Multi-Agent Reinforcement Learning with Domain-Randomization for Complex Analog Circuit Design Migration

Handa Sun[1,2], Zhaori Bi[1,*], Wenning Jiang[1,3,*], Ye Lu[1,4], Changhao Yan[1,2], Fan Yang[1,2], Wenchuang Hu[5], Sheng-Guo Wang[1,6], Dian Zhou[1,7,†], Xuan Zeng[1,2,*]

[1] State Key Laboratory of Integrated Chips and Systems; [2] School of Microelectronics; [3] Frontier Institute of Chip and System; [4] School of Information Science and Technology; Fudan University, Shanghai, China

[5] Precision Medicine Center of West China Hospital, Sichuan University, Chengdu, China

[6] Department of ET and Department of ECE, University of North Carolina at Charlotte, Charlotte, USA

[7] Department of Electrical Engineering, University of Texas at Dallas, Richardson, Texas, USA

ABSTRACT

Automated analog circuit design migration significantly alleviates the burden on designers in circuit sizing under various operating conditions. Conventional methods model the migration problem as black-box optimization, requiring excessive iterations of costly simulations to converge. Reinforcement learning exhibits significant promise in transfer learning, as it enables the generation of circuits that fulfill specifications efficiently. The paper proposes a novel value decomposition-based multi-agent reinforcement learning framework, aiming to model complex analog circuits and eliminate the need for manually defined specifications of sub-circuits for new operating conditions. Additionally, it incorporates domain randomization techniques to efficiently generate circuits that meet unforeseen scenarios with minimal simulations. Experiment demonstrates that our algorithm can efficiently generate circuits meeting specifications under new operating conditions in few number of steps, outperforming state-of-the-art methods.

KEYWORDS

Circuit Design Migration, Multi-agent Reinforcement Learning, Analog Circuit Synthesis

1 INTRODUCTION

The migration of analog circuit designs is crucial for promptly addressing diverse application scenarios without alterations to the circuit topology, but rather by adjusting the design parameters. In applications like an Analog to Digital Converter (ADC) circuit, variations in operating conditions, such as the sampling frequency, are common. However, analog circuit design is a time-consuming process, especially when there are changes in circuit conditions, often requiring designers to undergo redesign efforts once again, resulting in significant time wastage. Prevailing approaches formulates design migration as an optimization problem [5, 6], necessitating numerous iterations to achieve convergence. As analog blocks progress toward system-level integration, the escalating complexity and scale contribute to heightened intricacies in design, leading to an exponential increase in optimization costs.

We give a brief summary of circuit migration algorithms, categorizing them into two groups: optimization-based and reinforcement learning-based methods. Optimization-based methods employ an iterative approach, generating candidate designs to progressively converge towards the optimal point. The objective is to attain a design that fulfills all specifications while optimizing the circuits' objective. Various optimization techniques are introduced to address the design migration problem, including population-based optimization [9], gradient-based optimization [1, 11], and Bayesian optimization-based approaches [7, 15]. The convergence rate significantly depends on the circuit scales. In the case of complex analog circuits, the high-dimensional design parameters pose challenges in terms of computational complexity, which leads to the necessity of a large number of simulations to thoroughly explore and exploit the entire design space.

Reinforcement learning methods aim to train a model capable of generating a path leading to the optimized design. Recently, researcher proposes single agent reinforcement learning to optimize small scale analog circuit. GCN-RL [13] utilizes a graph convolutional neural network to model the relationship between analog circuit topology and circuit performance. This approach transfers design knowledge correlated with circuits of similar functions. Due to the growing computational complexity of graph neural network training with increasing circuit scales, GCN-RL encounter challenges in handling complex circuits. DNN-Opt [2] introduces a two-stage deep learning network as a surrogate model to enable efficient optimization. However, when dealing with migration, DNN-Opt still requires iterations to re-explore and re-exploit the design space, incurring significant costs.

Multi-agent reinforcement learning (MARL) employs multiple agents to partition a system-level circuit into sub-functional blocks, incorporating the intricate coupling effects between sub-blocks of circuits into the learning model [14]. This enables the representation of complex circuit behavior. However, the MARL method faces challenges in accurately and automatically decomposing the contribution between sub-blocks, which necessitates the manual

*Corresponding authors:{zhaori_bi,wenningjiang, xzeng}@fudan.edu.cn
†Emeritus Professor, University of Texas at Dallas .

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3656523

definition of specifications for sub-blocks of complex analog systems, relying on the design experience of human experts. When conditions change, the model cannot automatically and accurately adapt to the new operating condition. The transfer-ability of multi-agent reinforcement learning in analog design automation has not been thoroughly researched.

To address the aforementioned challenges, we introduce an innovative multi-agent reinforcement learning framework based on value decomposition. The proposed framework is crafted to evaluate the contribution of each sub-circuit to the overall system-level circuit performance. During the training stage, we employ a randomization strategy for the selection of training operating conditions to establish the transferability of the learning model. Upon successful training, the model demonstrates efficiency in producing designs that meet target specifications with a minimal number of simulations. This method facilitates the establishment of a fully automated reinforcement learning process for the intricate task of analog circuit design migration. Our principal contributions are outlined as follows.

- We propose an efficient value decomposition-based multi-agent reinforcement learning framework, which enables the establishment of a fully automated reinforcement learning process for complex analog circuit design migration. A mixing network is integrated to model the relationship between the system-level circuit and sub-blocks as a higher-level structure, thus automatically assembling the value function of each sub-block. The proposed structure demonstrates enhanced accuracy, automation, and robustness in system-level circuit design compared to the previous algorithm.
- We propose an innovative domain-randomization method for training reinforcement learning models to facilitate circuit design migration across various operating conditions. To enhance the transferability of the reinforcement learning model, we incorporate an additional random variable representing the operating condition. This augmentation enables the training process to learn the relationships under different operating conditions. To mitigate the need for an extensive amount of training data across all operating conditions, we introduce domain-randomization techniques, which involves sampling a specific number of training points only on selected operating conditions, thereby maintaining the model's generalizability.

2 METHODOLOGY

2.1 Problem Formulation

Conventional methods approach the circuit migration problem for new operating conditions as distinct tasks, requiring optimization from the beginning each time. In general, the circuit migration design problem is defined as equation(1).

$$\begin{aligned} &minimize \quad FOM(\boldsymbol{x}|\theta_k), \\ &s.t.\ c_i(\boldsymbol{x}|\theta_k) \leq 0, i = 1, ..., N \\ &\boldsymbol{x}_L \leq \boldsymbol{x} \leq \boldsymbol{x}_U, \boldsymbol{x} \in R^d, \end{aligned} \tag{1}$$

, where $\boldsymbol{x}$ denotes the design parameters and c_i denotes the constraint. Figures of Merit(FOM) and constraints are evaluated under the circuit operating condition(e.g., voltage, temperature, frequency).

Heuristically, knowledge from the circuit migration problem can be reused, we can define it as a self-adaptive generation problem. In the context of reinforcement learning, the migration problem pertains to the acquisition of knowledge necessary for generating design parameters that align with the target operating condition θ_k.

In this work, we define the universal Operating Condition Set (OCS) as $\Theta = \{\theta_i, i = 1...N\}$, where θ_i may denote operating frequency or voltage. The sets for training and migrating, denoted as Θ_p and Θ_t, respectively, are subject to the condition that $\Theta_t \cup \Theta_p \subseteq \Theta, \Theta_t \cap \Theta_p = \emptyset$. In the context of reinforcement learning, D represents the transitions (s, a, r, a') dataset generated by agent, $\mathbb{V}$ represents the learning process aimed at fitting a model M with network parameters W. During the training stage, we input the dataset D under the condition Θ_t, as defined in equation (2).

$$\mathbb{V}\left[D(\boldsymbol{x}, \boldsymbol{y})|\Theta_t\right] \longrightarrow M_t\left[W|\Theta_t\right] \tag{2}$$

Subsequently, during the migration stage, we modify the operating conditions to Θ_p over a few iterations, in order to create the adapted migration model M_p as specified in equation (3). It is during the migration stage that we identify the optimal design $\boldsymbol{x}^*, \boldsymbol{y}^*$ from $D'(\boldsymbol{x}, \boldsymbol{y})$ for the target operating condition θ_k.

$$\mathbb{V}\left[D'(\boldsymbol{x}, \boldsymbol{y})|(M_t, \theta_k)\right] \longrightarrow M_p\left[W'|\theta_k\right], \theta_k \in \Theta_p \tag{3}$$

2.2 The Overall Framework

The previously published MARL (Multi-Agent Reinforcement Learning) approach requires manual definition of specifications for each sub-circuit, presenting a significant obstacle for the automatic adaptation to new operating conditions. We propose a novel multi-agent reinforcement learning framework based on value decomposition to dynamically evaluate an agent's credit, reflecting the sub-circuit's contribution to the system-level circuit performance, enabling the establishment of a fully automated RL process for complex analog circuit design migration.

The proposed framework is shown in Figure 1, which encompasses the decomposition of the system-level circuit according to sub-circuit function, domain randomization of the circuit's operating conditions, and the training and migration process of Value decomposition-based MARL. **(I) System-level circuit decomposition**. In order to model complex circuits, we partition the system-level circuit into multiple sub-circuits based on their functions, and allocate an agent to each sub-circuit. As illustrated in Figure 1(a), an 8-bit ADC is represented by four agents, each corresponding to the Driver, Comparator, DAC logic, and Comparator logic. **(II) Value decomposition-based MARL**. To achieve a fully automated RL process for migrating complex analog circuit designs, we propose a circuit operating condition-adaptive value decomposition-based MARL allowing for the dynamic assessment of each individual agent's credit and aims to optimize the overall performance of the system-level circuit, which is demonstrated in Figure 1(b). **(III) Domain randomization**. To improve the adaptability to complex migration scenarios across various circuit operating conditions, we propose a domain randomization-based migration approach to perform the operating conditions randomization during EVDMARL training, which is shown in Figure 1(e).

During the EVDMARL training stage, the following steps are undertaken. **(I) Initialization**. This includes randomly generating an

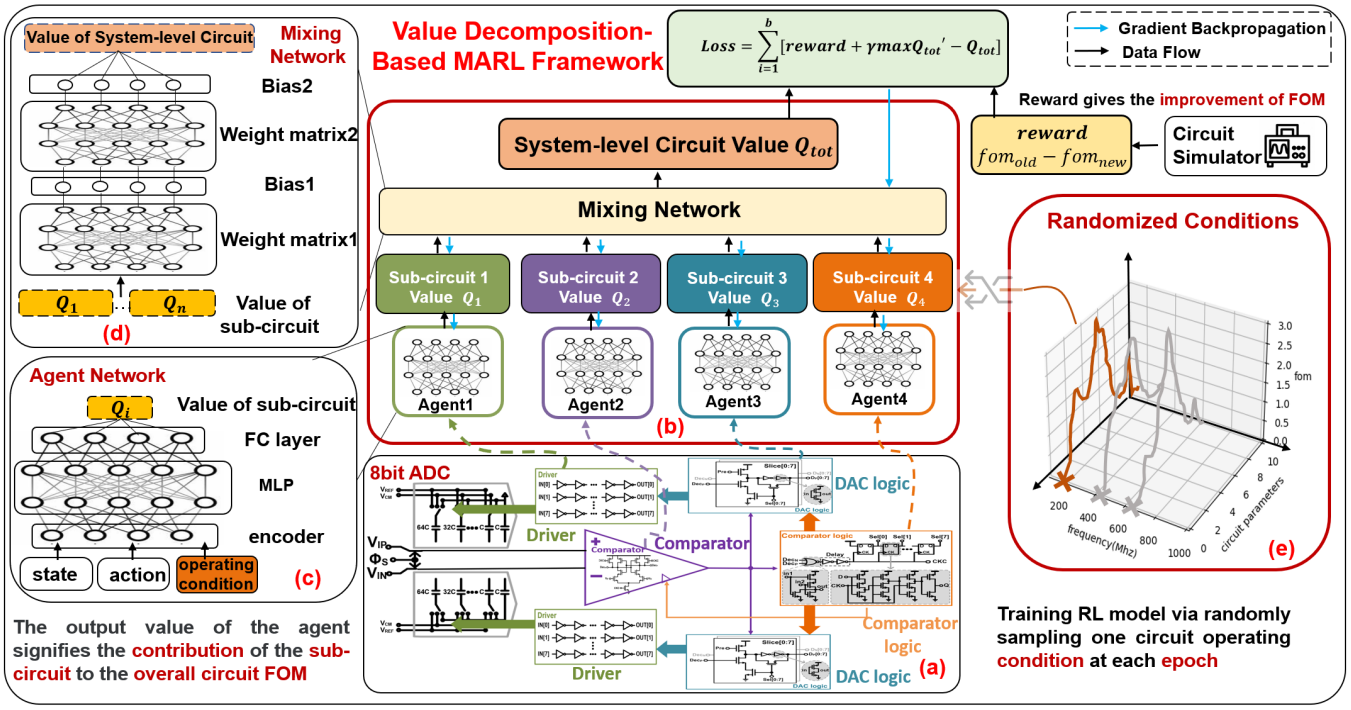

Figure 1: (a) *System-level circuit decomposition.* A complex circuit is partitioned into sub-circuit using topology information and circuit function. (b) *The Value decomposition-based MARL.* Each agent generates the value for the sub-circuit and feeds it into the mixing network, which yields the value of the system-level circuit. The system-level circuit value, along with the FOM improvement (i.e., reward), is calculated as the loss, and subsequently transmitted back to the neural network for backpropagation. (c) *Agent Network.* Agent network takes the state and possible actions as input, output is the value of sub-circuit. The structure includes an encoder, a multi-layer perceptron (MLP), and a fully connected (FC) layer. (d) *Mixing Network.* The mixing network takes the value of each sub-circuit as input and generates the value of the entire circuit. Its weight matrix and bias are generated by a hypernetwork using an absolute activation function, ensuring that all parameters are positive. (e) *Domain randomization.* Domain randomization is utilized to introduce random operating conditions during EVDMARL training, thereby improving the adaptability of the algorithm.

operating condition and a set of random design parameters, as well as initializing the agent network and mixing network. **(II) Agent action**: Each agent suggests *Actions* to the corresponding sub-circuit parameters based on the current circuit parameters, defined as *State*, using the cross-entropy method(CEM)[3] in algorithm 2. *State* of the overall circuit is then updated based on the actions taken by the agents. **(III) Value calculation**: Each agent network evaluates its sub-circuit *Value*, which is then sent to the mixing network for the computation of the system-level circuit *Value*. **(IV) Back propagation**: The circuit's *State* is sent to the simulator, and a *Reward* is provided. This reward is used to calculate the Temporal Difference (TD) error, which serves as the loss function and is utilized for backpropagation to train each network. Repeat steps (I) to (IV) in each epoch until EVDMARL model converges. The overall training algorithm is listed in Algorithm1.

During the migration stage, the agent suggests *Actions* based on the network trained in the training stage, and data generated under new circuit conditions is fed into the neural network to fit and adapt to the new operating conditions. Mixing networks can also automatically adjust their models to changes in operating conditions with minimal steps. Key concepts related to EVDMARL are listed as below.

State. State refers to a vector that represents system-level circuit design parameters, including transistor width and channel length etc. denoted as follows. $[W_1, L_1, W_2, L_2, ...W_n, L_n]$, where W_i represents the width of the i-th transistor and L_i represents the channel length of the i-th transistor. The incorporation of the global state enables each agent to take into account the actions of other agents, leading to improved decision-making capabilities.

Action. Action denotes a vector that represents the adjustments made by each agent to the sub-circuit design parameters. For example, $\left[\triangle W_1^a, \triangle L_1^a, \triangle W_2^a, \triangle L_2^a, ... \triangle W_n^a, \triangle L_n^a\right]$ where $\triangle W_i^a$ represents the change of the i-th design parameter of agent a.

Reward. Reward is calculated based on the changes in the FOM resulting from the actions, reflecting the improvement in the performance of the system-level circuit. The one-step reward is defined as equation 4.

$$reward = \begin{cases} +1 & fom_{new} < fom_{last} \\ -1 & fom_{new} > fom_{last} \end{cases} \tag{4}$$

When a specific specification is achieved within a single step, a substantial reward (e.g., +10) is provided to encourage the agents to explore this area. In the context of migration problems, we steer clear of rewards that directly mirror the actual improvement in

specification values, since these specifications can fluctuate with changes in circuit operating conditions. Hence, we utilize a normalized reward formulation.

Value. Value is defined as the expected future reward of taking a certain action conditioned on the certain circuit state. The value is defined as equation 5.

$$Q^{\pi}(\mathcal{S}_t, \mathcal{A}_t) = \mathbb{E}_{s_{t+1:\infty}, a_{t+1:\infty}}\left[R_t \mid \mathcal{S}_t, \mathcal{A}_t\right] \tag{5}$$

, where $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ is the cumulative reward from time t with discount factor γ^i. At each time-step, we choose the action with the highest value to maximize the potential reward, aligning with the goal of optimizing the FOM of the circuit, as shown in equation 6.

$$\mathcal{A}_t^{a*} = \underset{\mathcal{A}_t^a}{\operatorname{argmax}}(Q^a(\mathcal{S}_t, \mathcal{A}_t^a|\theta_i)) \tag{6}$$

Algorithm 1: EVDMARL training algorithm

Initialize: agent network, mixing network and hyper-network
for *iteration* $i = 1$ *to* N **do**
  Initialize a random state: $\mathcal{S}_0 \leftarrow$ state space;
  Randomly generate an operating condition: θ_i;
  for *step* $t = 1$ *to* T **do**
    for *agent* $a = 1$ *to* M **do**
      Choosing action using CEM method: $\mathcal{A}_t^{a*} = \underset{\mathcal{A}_t^a}{\operatorname{argmax}}(Q^a(\mathcal{S}_t, \mathcal{A}_t^a|\theta_i))$;
    end
    Calculating sub-circuit value using agent network: $Q_t^a = Q^a(\mathcal{S}_t, \mathcal{A}_t^{a*}|\theta_i)$;
    Calculating system-level circuit value using mixing network: $Q_t^{\text{tot}} = \text{mixing}(Q_t^1, ..., Q_t^m)$;
    Updating system-level circuit state: $\mathcal{S}_{t+1} = \mathcal{S}_t + \mathcal{A}_t$;
    Simulator gives new FOM and reward r=reward;
    Training networks end-to-end using TD error: $\mathcal{L}(\theta) = [r + \gamma \underset{\mathcal{A}_{t+1}}{max} Q^{tot}(\mathcal{S}_{t+1}, \mathcal{A}_{t+1}|\theta_i) - Q^{tot}(\mathcal{S}_t, \mathcal{A}_t|\theta_i)]^2$;
  end
end

2.3 Value-decomposition Based MARL for Analog Design Automation

Representing the value of a system-level circuit using analytical expressions is a challenging task, leading to the adoption of neural networks for modeling purposes. In multi-agent reinforcement learning, modeling the value function of the system-level circuit is challenging due to the extensive action space involved. Inspired by QMIX [12] algorithm, we propose EVDMARL framework, where individual agent network is to model the value function of each sub-circuit, while a mixing network combines these individual value functions to model the value function of the circuit. The agent network evaluates whether a change in design parameters is beneficial or detrimental by generating sub-circuit values as outputs. The mixing network assesses the overall improvement of the circuit based on the system-level circuit value. The accuracy and reliability of these evaluations are refined through training and correction using rewards provided by the simulator. Once trained, the agent network and mixing network can effectively assess the impact of parameter changes on circuit performance and select the optimal action based on the circuit's current state to optimize design parameters. This innovative approach enables a more comprehensive representation of the circuit's value and enhances the level of automation within the system.

Agent Network. Each agent is an artificial neural network composed of one input encoder, one output layer and 5 hidden layers. They take the state of the whole circuit and possible actions of corresponding sub-circuit as input, output is the value to take such action in such state for the agent. To determine the optimal action based on the current circuit state, we utilize the cross-entropy method(CEM) [3] to sample an action from the sub-circuit action space, which has the highest output value from the agent network. This indicates that this action is the best choice given the conditions of the circuit at that moment. The CEM algorithm is listed as algorithm 2.

Algorithm 2: CEM algorithm

Initialize the random variable $\mathcal{A}^a$ with a multivariate normal distribution:$P_0(\mathcal{A}^a) = \mathcal{N}(\mu, \Sigma)$;
for *iteration* $i = 1$ *to* M **do**
  initialize buffer $B = \{\}$;
  for *sample* $e = 1$ *to* N **do**
    sample $\mathcal{A}_e^a \sim P_i(\mathcal{A}^a)$;
    calculate value of such action: $Q_e^a = Q^a(\mathcal{S}, \mathcal{A}_e^a|\theta)$;
    store $\mathcal{A}_e^a, Q_e^a$ in B;
  end
  $\hat{B}$=select N_{best} top samples ranked by Q_e^a in B;
  calculate the mean and variance of samples in $\hat{B}$;
  update μ, Σ of distribution $P_i(\mathcal{A}^a)$;
end
sample an action from distribution $P_M(\mathcal{A}^a)$;

Mixing Network. Considering that the design goal of each sub-circuit is to optimize the FOM of the system-level circuit, the relationship between agents can be modeled as a purely cooperative relationship, that is, the system-level circuit value is positively related to the value of each agent. The relationship between the value of the sub-circuit and the value of the overall circuit should satisfy the constraint as equation. 7.

$$\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \forall a \in A \tag{7}$$

To ensure a positive relationship between the value of the sub-circuit and the value of the system-level circuit, we introduce a mixing network with a hyper-network architecture that draws inspiration from QMIX [12] , where the weight parameters of the mixing network need to be positive. A hyper-network which has an absolute activation function is used to generate the weight parameters and bias of the mixing network.

2.4 Domain Randomization technique

The goal of our migration problem is to find a proper searching policy when the circuit comes to a new operating condition. We propose a generalization method based on Domain randomization technique[10] to enhance the ability of migration. Since Q-learning

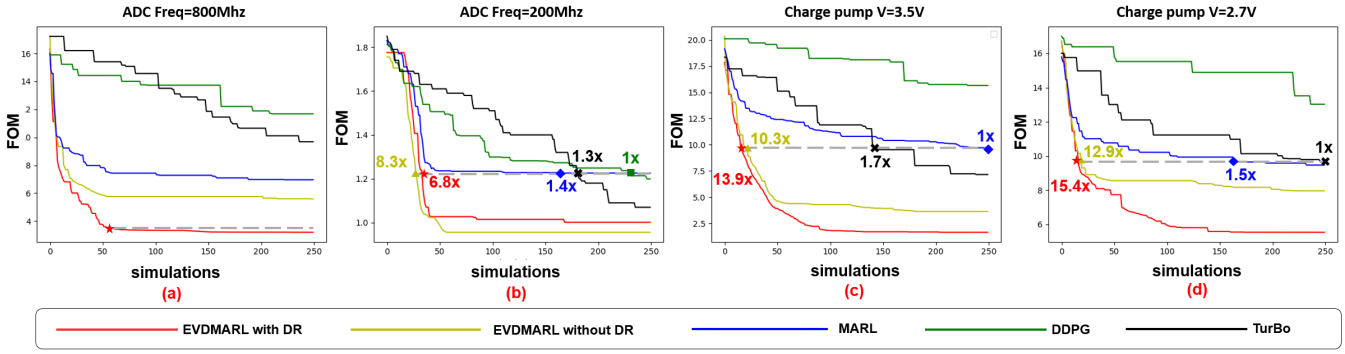

Figure 2: (a) Result of 8 bit ADC(800MHz). EVDMARL with DR meets the specifications within 250 simulations. (b) Result of 8 bit ADC(200MHz). EVDMARL achieves the best FOM of 1.23 and is 8.3 times faster than DDPG. (c) Result of Charge pump(3.5V). All algorithms except DDPG meet the spec, EVDMARL with DR reaches the best FOM of 9.71 and is 13.9 times faster than MARL. (d) Result of Charge pump(2.7V). EVDMARL with DR meets spec within 16 simulations and achieves the best FOM of 9.69.

is an off-policy method, we can populate the training database with data from diverse circuit operating conditions. This helps us to train a more comprehensive and operating condition-awared policy. Our approach conditions the searching policy on the circuit operating condition, which is treated as a random variable as equation 8.

$$\pi_{tot} = \pi(a_t|s_t, \theta) \tag{8}$$

By incorporating this variability into the search process, we can improve the adaptability of the policy to different circuit operating conditions, leading to better overall performance. During each training epoch, we randomly initialize an operating condition and generate a design trajectory based on this condition. This trajectory data is then incorporated into the training database. By enabling the reinforcement learning network to converge to different operating conditions, we expect it to possess the ability to generalize to unseen operating conditions.

3 EXPERIMENTS

3.1 8 Bit ADC

We design a complex system-level ADC circuit, whose schematic is depicted in Figure1 (a). In the process of designing an ADC circuit, designers usually have to create circuits for different frequency points. As they encounter new frequency points, they need to adjust the design parameters, resulting in a complex and iterative process. Designing for higher frequency points is often more difficult, so designers typically begin with higher frequencies and then ensure compatibility with lower frequencies.

To address this challenge, we trained our algorithm using domain randomization within the 400-600MHz range, and then utilized it to efficiently generate designs at 200MHz and 800MHz. In the ADC design process, four sub-circuits - the Driver, Comparator, DAC logic, and Comparator logic - need to be carefully designed. To optimize the design, we assigned an agent to each sub-circuit. The optimization objectives and constraints are listed as equation 9

$$\begin{aligned} &minimize \quad \frac{P}{2^{ENOB} \times f_s} \\ &s.t. \quad THD < -49.5dB \end{aligned} \tag{9}$$

,where $ENOB$ denotes the Effect Number of Bits, and THD denotes the Total Harmonic Distortion. For each case, we compare the proposed EVDMARL method with TuRBO[4], DDPG[8] and MARL[14]. The results are showed in Figure 2 and Table 1. At 800MHz, the EVDMARL with DR algorithm is the only method that satisfies the specifications within 250 simulations. At 200MHz, while all algorithms meet the design specifications, our approach achieves a speed 6.8-8.3 times faster than the slowest algorithm. These results show that at the new operating frequency, our algorithm requires a minimum of only 34 simulations to generate a well-designed ADC circuit that meets the design specifications.

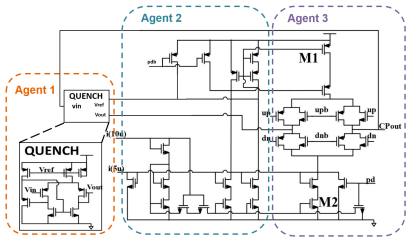

Figure 3: Schematic of Charge pump.

3.2 Charge pump

The Charge pump, as shown in Figure 3, is highly sensitive to changes in input voltage. When the input voltage undergoes a change, the design parameters require a redesign. We conducted training for each algorithm within the 2.9V-3.3V range and subsequently generated design parameters at 3.5V and 2.7V. At 3.5V, all algorithms except DDPG fulfill the design requirements. Among them, the proposed EVDMARL with DR algorithm achieves the highest speed, being 13.9 times faster than MARL, and attains the best FOM of 1.66. At 2.7V, our method demonstrates a speed 15.4 times faster than the slowest algorithm, while also achieving the best FOM of 5.54. The experiments demonstrate that our algorithms can efficiently generate a Charge pump circuit under the newly defined operating conditions, meeting the specifications within only a minimum of 16 simulations.

Table 1: Experiment results.

Circuits	Algorithm	FOM	Sim. No.	Speed up	Circuits	Algorithm	FOM	Sim. No.	Speed up
8 bit ADC(800MHz)	TuRBO	9.68 (±0.08)	250	fail	8 bit ADC(200MHz)	TuRBO	1.23 (±0.03)	181	1.3×
	DDPG	11.68 (±0.51)	250	fail		DDPG	1.23 (±0.05)	231	1×
	MARL	6.96 (±0.04)	250	fail		MARL	1.23 (±0.03)	164	1.4×
	EVDMARL	**5.59 (±0.02)**	**250**	**fail**		**EVDMARL**	**1.23 (±0.01)**	**28**	**8.3×**
	EVDMARL with DR	**3.46 (±0.03)**	**60**	–		**EVDMARL with DR**	**1.23 (±0.02)**	**34**	**6.8×**
Charge pump(3.5V)	TuRBO	9.71 (±0.02)	143	1.7×	Charge pump(2.7V)	TuRBO	9.69 (±0.04)	246	1×
	DDPG	15.65 (±0.14)	250	fail		DDPG	13.03 (±0.11)	250	fail
	MARL	9.71 (±0.01)	236	1×		MARL	9.69 (±0.01)	164	1.6×
	EVDMARL	**9.71 (±0.03)**	**23**	**10.3×**		**EVDMARL**	**9.69 (±0.02)**	**19**	**12.9×**
	EVDMARL with DR	**9.71 (±0.02)**	**17**	**13.9×**		**EVDMARL with DR**	**9.69 (±0.02)**	**16**	**15.4×**

3.3 Result Discussion

To summarize, our algorithm is up to 15.4 times faster than TuRBO in migrating applications, because we utilize information from other circuit operating condition, while TuRBO treats new operating conditions as entirely new problems. Our algorithm is also up to 13.9 times faster than the single-agent DDPG algorithm and achieves higher FOM in a shorter time. This is because it excels in modeling complex circuits, enabling efficient design generation for high-dimensional circuits. In contrast, the DDPG algorithm often faces convergence challenges during training and is not effective in addressing migration problems. Our algorithm always achieves better FOM than MARL for the reason that it models the individual contributions of sub-circuits to the system-level circuit, allowing for adaptive and precise value decomposition and quick adaptation to different operating conditions. In addition, EVDMARL algorithm combined with Domain randomization technology achieves better performance because domain randomization Technology enhances EVDMARL's generalization ability. Experimental results show that the proposed EVDMARL method can generate circuit designs that meet specifications rapidly while achieving superior performance compared to existing methods, all within a shorter time frame.

Furthermore, although we exclusively conduct experimental validation on the operating conditions unseen during training, EVDMARL can also migrating to observed conditions during training rapidly. We stress that such migration is also meaningful. As real-world operating conditions are always continuous, leveraging DR Technology to acquire knowledge across these continuous points significantly reduces the time required compared to addressing each operational point individually.

4 CONCLUSION

In this paper, a novel value decomposition-based MARL with domain-randomization for complex analog circuit design migration is proposed to generate circuit with new operating conditions efficiently and automatically. Experimental results on 8 bit ADC and Charge pump demonstrate that our algorithm outperforms existing methods by rapidly generating circuit designs that meet specifications within at least 17 simulations. For future research, the EVDMARL architecture for analog circuit design can be combined with other generalization methods such as Graph Neural Networks(GNN), meta-learning technique, and Transformer architecture, to improve the migrating ability.

5 ACKNOWLEDGMENTS

This research is supported partly by National Natural Science Foundation of China (NSFC) research projects 62141407, 62304052, 92373207, 62090025, 62350610270 and Shanghai Science and Technology Innovation Fund 23YF1402300.

REFERENCES

[1] Zhaori Bi, Dian Zhou, Sheng-Guo Wang, and Xuan Zeng. 2017. Optimization and quality estimation of circuit design via random region covering method. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 23, 1 (2017), 1–25.

[2] Ahmet F Budak, Prateek Bhansali, Bo Liu, Nan Sun, David Z Pan, and Chandramouli V Kashyap. 2021. Dnn-opt: An rl inspired optimization for analog circuit sizing using deep neural networks. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1219–1224.

[3] P. T. de Boer, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. 2005. A Tutorial on the Cross-Entropy Method. *Annals of Operations Research* 134, 19–67.

[4] David Eriksson, Michael Pearce, Jacob R Gardner, Ryan Turner, and Matthias Poloczek. 2019. Scalable Global Optimization via Local Bayesian Optimization. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Article 493, 12 pages.

[5] Morteza Fayazi, Zachary Colter, Ehsan Afshari, and Ronald Dreslinski. 2021. Applications of artificial intelligence on the modeling and optimization for analog and mixed-signal circuits: A review. *IEEE Transactions on Circuits and Systems I: Regular Papers* 68, 6 (2021), 2418–2431.

[6] Georges GE Gielen and Rob A Rutenbar. 2000. Computer-aided design of analog and mixed-signal integrated circuits. *Proc. IEEE* 88, 12 (2000), 1825–1854.

[7] Tianchen Gu, Wangzhen Li, Aidong Zhao, Zhaori Bi, Xudong Li, Fan Yang, Changhao Yan, Wenchuang Hu, Dian Zhou, Tao Cui, et al. 2023. BBGP-sDFO: Batch Bayesian and Gaussian Process Enhanced Subspace Derivative Free Optimization for High-Dimensional Analog Circuit Synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2023).

[8] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *CoRR* abs/1509.02971.

[9] Po-Cheng Pan, Hung-Ming Chen, Chien-Chih Lin, et al. 2013. PAGE: parallel agile genetic exploration towards utmost performance for analog circuit design.. In *DATE*. 1849–1854.

[10] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 3803–3810. https://doi.org/10.1109/ICRA.2018.8460528

[11] Liuxi Qian, Zhaori Bi, Dian Zhou, and Xuan Zeng. 2014. Automated technology migration methodology for mixed-signal circuit based on multistart optimization framework. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23, 11 (2014), 2595–2605.

[12] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2020. Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning, Vol. 21. JMLR.org.

[13] Hanrui Wang, Kuan Wang, Jiacheng Yang, Linxiao Shen, Nan Sun, Hae-Seung Lee, and Song Han. 2020. GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[14] Jinxin Zhang, Jiarui Bao, Zhangcheng Huang, Xuan Zeng, and Ye Lu. 2023. Automated Design of Complex Analog Circuits with Multiagent based Reinforcement Learning. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1109/DAC56929.2023.10247909

[15] Aidong Zhao, Xianan Wang, Zixiao Lin, Zhaori Bi, Xudong Li, Changhao Yan, Fan Yang, Li Shang, Dian Zhou, and Xuan Zeng. 2023. cVTS: A Constrained Voronoi Tree Search Method for High Dimensional Analog Circuit Synthesis. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.