Fast Design Space Adaptation with Deep Reinforcement Learning for Analog Circuit Sizing

Kai-En Yang*, Chia-Yu Tsai†, Hung-Hao Shen†, Chen-Feng Chiang†, Feng-Ming Tsai†,
Chung-An Wang†, Yiju Ting†, Chia-Shun Yeh†, Chin-Tang Lai†
*EECS, National Tsing Hua University, Hsinchu, Taiwan
†MediaTek Inc., Hsinchu, Taiwan

***Abstract*—We present a novel framework for design space search on analog circuit sizing using deep reinforcement learning (DRL). Nowadays, analog circuit design is a manual routine that requires heavy design efforts due to the absence of automation tools, motivating the urge to develop one. Prior approaches cast this process as an optimization problem. They use global search strategies based on DRL with complex network architectures. Nonetheless, the models are hard to converge and neglected various working conditions of PVT (process, voltage, temperature). In this work, we reduce the problem to a constraint satisfaction problem, where a local strategy is adopted. Thus, a simple feed-forward network with few layers can be used to implement a model-based reinforcement learning agent. To evaluate the value of the our framework in production, we cooperate with R&Ds in an IC design company. On circuits with TSMC advanced 5 and 6nm process, our agents can deliver PPA (performance, power, area) beyond human level. Furthermore, the product will be taped out in the near future.**

***Index Terms*—transistor sizing, artificial intelligence, reinforcement learning, electronic design automation**

I. Introduction

Consumer electronics over the years have undeniable impacts on almost every aspect of our daily lives, let alone the annual increment of computing power described by Moore's law, pioneering unprecedented possibilities. This remarkable progress has been accompanied by a collinearity with tremendous increases in chip design complexity.

Despite the majority of the mixed signal SoC area is occupied by digital circuitry, analog circuits are still essential for the chips to be able to communicate with and sense the rest of the world. However, the design effort of the analog counterpart is more onerous due to the heavy requirements of human expertise, and the lack of automation tools similar to digital subsystems.

One of the labor intensive task in analog design is transistor sizing. Currently, it is mostly done by labouring trial-and-error. The designers begin by applying their knowledge about the characteristics of transistors to select a reasonable range of candidate solutions, next explore the space with grid search. Then, get feedback from SPICE circuit simulations. Afterward, repeat the procedure until the specifications are met. The main challenge of automating sizing is that it has a very large space of potential solutions[1], which prior arts suffered from convergence problems and poor scalability[2].

In this paper, we propose a learning-based generalizable search framework (Fig. 1) to help increase R&D productivity during analog front-end sizing task (Fig. 2). Experiment results demonstrate that our agents can efficiently and effectively accomplish search in state-of-the-art designs with superior performance while achieving area enhancement over human. The contributions of our work are severalfold on different levels.

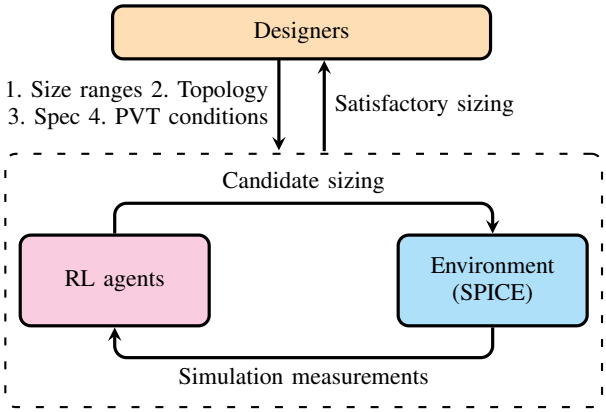

Fig. 1. Framework architecture

- **System level** We propose a general framework for IC design space search. The standardized API allows fast migration provided well-formulated problems. The ultimate goal of the framework is to increase R&D productivity with minimal extra efforts required.
- **Algorithm level** The proposed model-base reinforcement learning approach is different from prior methodologies. This novel design enables a more adequate implementation for usages in the industry.
- **Verification level** Considerations on PVT conditions better suits empirical needs. Previous negligence in multiple working conditions of a chip prohibits an approach from serviceable.

II. Prior arts

Various attempts have been presented to tackle this problem, and the progress follows the development of artificial intelligence.

The first success in artificial intelligence was expert system, a method that explicitly capture human knowledge and transfer them into computer programs[3].While the time complexity is a constant, a large amount of time is needed to build the system

and can only provide an initial trial, which is infeasible in the fast technology advancement.

Sizing automation could be framed as an optimization problem. Bayesian optimization (BO) is a popular choice because of the sample efficiency in finding the global optimum[4][5]. Despite of the promising results, the scalability is a major drawback. Note that the scalability addressed is the cubical increment of the number of samples, rather than the dimensionality of the space.

Recently introduced methods primarily leverage the current success in deep learning. Well-developed supervised learning methods, e.g.: image classification[6][7], object detection[8][9], were established as not suitable for IC design space optimization caused by the difficulty of providing sufficient data samples for training[2][10]. Some models demand re-training when a new set of specifications is assigned, hence, cannot be reused[11].

In later publications, model-free reinforcement learning comes prominence. AutoCkt[12] aims to train an efficient agent to explore and gain knowledge about the design space. The agents will then be used to generate trajectories during inference. Yet in practice, it is rarely necessary to traverse the space.

L2DC[13] exploits sophisticated sequence-to-sequence modeling using an encoder-decoder technique. GCN-RL[14] employs the latest innovation - graph convolutional neural networks - to learn features from the structures. This enables the model to reach better transferability between nodes and topologies. In spite of their ability to exceed human-level performance, the amount of human-engineering efforts in observation selection, network architecture design, and reward engineering reduce the feasibility of becoming a generalizable automation tool.

III. Problem formulation

Transistor sizing (Fig. 2) is an iterative process to determine a suitable set of lengths, widths, and multiplicities for each transistor in the topology in order to achieve the desired specifications. This scheme is often regarded as a trade-off between performance, power, and area (PPA). Larger transistor sizes normally lead to greater performance, but consume more power and area.

Analog sizing can be formulated as a multi-objective constrained optimization problem, defined in (1).

$$\begin{aligned} \text{Minimize} \quad & F_{m,c}(X) && m = 1, 2, ...N_m \\ & && c = 1, 2, ...N_c \\ \text{subject to} \quad & C_{d,c}(X) < 0 && d = 1, 2, ...N_d \\ & && c = 1, 2, ...N_c \\ & X \in \mathcal{D}_s \end{aligned} \tag{1}$$

where X is a vector of variables to be optimized; $\mathcal{D}_s$ is the design space; $F_{c,m}(X)$ is the m^{th} objective function under the c^{th} PVT condition; $C(X)$ is the d^{th} constraint under the c^{th} PVT condition.

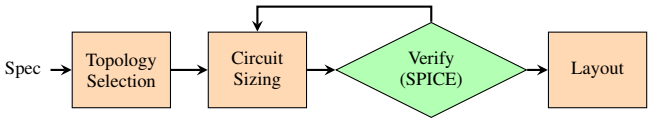

Fig. 2. Analog circuit design flow

IV. Proposed framework

The proposed framework is shown in Fig. 1. Deep reinforcement learning is believed to be a robust methodology for solving combinatorial search problem in various disciplines without human in the loop, such as games[15][16][17], robotic control[18][19], neural architecture search[20][21], and IC design[22][23]. We thus cast the transistor sizing task as a DRL framework. This allows us to automate the process, while being able to adapt to the environment fast based on past experiences, and evolve over time. This system consists of several subsystems described in the following sections.

A. Agents

Transistor sizing (Fig. 2) has long been formulated as an optimization problem, in which agents and optimizers are implemented to search for optimal points in objective functions. However, it is worth a rethink of what is an adequate implementation to integrate with the flow of a R&D.

Under the pressure of time-to-market, particularly during fast technological advances, meeting specifications is already challenging, needless to say finding the best solution. In other words, a set of satisfactory parameters is desired over a perfect one.

In this manner, the problem can be reduced to a constraint satisfaction problem (CSP), where the objective function is replaced as a 0 constant function (2).

$$\begin{aligned} \text{Minimize} \quad & 0 \\ \text{subject to} \quad & C_{d,c}(X) < 0 && d = 1, 2, ...N_d \\ & && c = 1, 2, ...N_c \\ & X \in \mathcal{D}_s \end{aligned} \tag{2}$$

More generally, CSP is defined as a triple $\langle X, D, \mathcal{C} \rangle$ (3).

$$\begin{aligned} X = \quad & \{x_1, x_2, ..., x_n\} \\ \mathcal{D} = \quad & \{D_1, D_2, ..., D_n\}, \quad D_i = \quad \{b_1, b_2, ..., b_l\} \\ \mathcal{C} = \quad & \{C_1, C_2, ..., C_k\}, \quad C_j = \quad (t_j, r_j) \end{aligned} \tag{3}$$

where X is a set of finite sizing variables to be searched, each has a non-empty domain D_i, namely the design space, and b_k is the possible values. C is a set of constraints. A constraint is a pair consists of a constraint scope t_j and a relation r_j over the variables in the scope, limiting feasible permutations of assignments. In our case, the relation cannot be explicitly expressed since it is the complex computation inside of the SPICE simulation, denoted as S_{pice} function.

With this, the intention is to find any complete assignment that is consistent. This also prevents over designing the circuit. Therefore, an algorithm targeting on fast exploring feasible

solutions instead of a global one could make more merit. An akin method is meta-learning[24]. The model attempts to adapt to new tasks quickly, rather than focusing on a specific environment.

An effective realization for solving CSPs is local search[25], which introduces appealing three-fold advantages.

- **Faster environment adaptation** By reducing the domain to local allows fewer iterations for initialization. In addition, the circuit space is locally continuous. By traversing the area around a known optimal point, an interesting observation is that neighboring points show similar optimality.
- **The use of model-based agents with supervised learning** Related works criticized supervised learning as per the metric of global goodness of fit. However, it still works given that the local landscape can be captured. Moreover, no reward is involved in model-based agents, making it insensitive to reward engineering.
- **Easier implementation and convergence** State-of-the-art model-free agents behave based on a surrogate network modeling the relationship between past trajectories and the next actions to take. However, this custom model tends to complicate the problem which makes it is hard to converge. Whereas the training routine of supervised learning is well-studied and relatively trivial.

All virtues combined actuate a model-based approach, where a direct modeling of the circuit space from transistor sizes to circuit measurements is mapped, imitating the behavior of a SPICE simulator. A simple feed-forward neural network with 3 layers can be used as a SPICE function approximator.

Model-based agent is an obscured choice until a recent novel publication[17] that use it in Atari game play. They claimed that humans can learn fast thank to the intuitive understanding of the physical processes, and we can apply it to predict the future. Thus, agents who possess such skill could be more sample efficient.

Each search starts by a random exploration in the design space. Next, the most optimal point will be selected as the local area to dive in. The policy is that by modeling the local landscape, a candidate solution can be chosen as the next step based on the expected value computed with predicted measurements. Even though an optimization or another policy network can be incorporated to find the candidate with the maximum value, to take the advantage of the fast inference time of a neural network, a more vanilla Monte Carlo sampling is used. The pseudocode of the realization is detailed in Algorithm 1.

Algorithm 1 Fast local explorer algorithm

1: initialize $trajectory \leftarrow [\,]$
2: initialize $\tilde{S}_0 \leftarrow (X^1, X^2, ..., X^N)$ is N samples of X
3: evaluate performance via $S_{pice}(\tilde{S}_0)$
4: compute values V_0 of $S_{pice}(\tilde{S}_0)$ with V_{alue} function
5: find best $a_0 \leftarrow \text{argmax}_{\tilde{S}_0 \in \mathcal{D}^N} V_0$
6: **if** $\tilde{S}_0 \in \mathcal{C}$ **then**
7: $\quad return\ (a_0, S_{pice}(a_0))$
8: **else**
9: $\quad$ reduce domain to local $\mathcal{D}_L \leftarrow R_{eduction}(a_0, \mathcal{D})$
10: **end if**
11: $trajectory.append((\tilde{S}_0, S_{pice}(\tilde{S}_0)))$ for training
12: initialize $i \leftarrow 0$
13: **while** True **do**
14: $\quad i \leftarrow i + 1$
15: $\quad NN.train(trajectory)$
16: $\quad$ sample m Xs $\tilde{S}_i = (X^1, ..., X^m)$, $X_j \in \mathcal{D}_L$
17: $\quad$ estimate performance via $\hat{y}_i \leftarrow NN.predict(\tilde{S}_i)$
18: $\quad$ compute values V_i of $\hat{y}_i$ with V_{alue} function
19: $\quad$ select $a_i \leftarrow \text{argmax}_{\tilde{S}_i \in \mathcal{D}_L} V_i$
20: $\quad$ **if** $\tilde{S}_i \in \mathcal{C}$ **then**
21: $\quad\quad return\ (a_i, S_{pice}(a_i))$
22: $\quad$ **end if**
23: $\quad trajectory.append((a_i, S_{pice}(a_i)))$
24: $\quad$ **if** $i > C_{riterion}$ **then**
25: $\quad\quad$ escape, jump to line 1
26: $\quad$ **end if**
27: **end while**

B. Reward (value) engineering

For our agents, there is a value function to estimate the merit of a set of circuit measurements after SPICE simulation. This is served as an indication of where to go next. Unlike model-free actor-critic methods, values do not participate in training. Consequently, the design of the formula does not affect the convergence of the model.

In the spirit of simplicity and generalization, we utilize a naive tactic where the value is the sum of normalized measurements. This way, no extra information is acquired. However, in terms of the trade-off between PPA, a second stage value function could be implemented to explicitly encode the importance of each measurement once the agent enters a optimal local area.

C. PVT exploration strategy

In order to push the system to production, one important aspect to consider is the PVT (process, voltage, temperature) conditions. To guarantee that a chip is able to work under the variations of fabrications, power supplies, and environments, a number of corners have to be signed off before tape-out. Nonetheless, to best of our knowledge, no prior art has accommodated such regime.

A simple way to explore the conditions would be to test all PVTs every time a new set of assignments is obtained. Yet for deployment, this strategy would be posed a waste in computing resources and EDA licenses if the agent is not yet ready for verification. Another intuitive approach is by accumulating rewards in each PVT. One limitation is that if the reward is implemented relative to previous steps, the summation of the rewards would be irrational.

Inspired by heuristics of designers, first focus the search on a single condition, usually the most difficult condition

to search, assuming that by overcoming the hardest PVT, easier ones would be easy. Once all the specifications are met, verifications will be conducted to confirm that the set of assignments are legal under all other conditions as well. Accordingly, the proposed progressive strategy is that if the initial condition does not apply, the corner with the least value will be chosen to be searched next, until all constraints are satisfied under all conditions (Fig. 3). If multiple corners are under search, the candidate solution will be taken as the complete assignments with the lowest expected value. Note that each PVT condition has its own independent model.

In Algorithm. 3, the search is demonstrated in a uniform condition. However, it is painless to generalize to our progressive strategy.

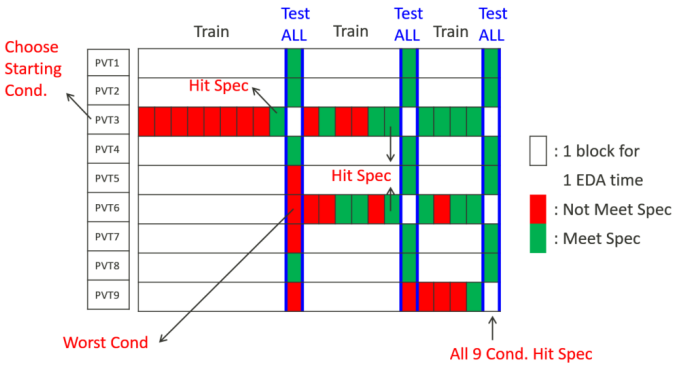

Fig. 3. Progressive PVT exploration strategy

D. API

A framework with a powerful search algorithm could be wasteful if the user interface is unfriendly or requires extra effort to use. In the introduced API, only information that the designers need in their original flow is acquired. In particular, human experts will need to identify the transistor sizes to tune, the ranges of variables, the circuit topology, the measurements to observe from SPICE simulations, and the specifications for each corner. Once the configuration is set, an automatic script will construct necessary components, namely the neural network architectures and hyperparameters of the network, which are also dynamically scheduled on the fly. That is to say, our framework is equivalent to a SPICE decorator where the DRL agents are encapsulated, which seamlessly automates transistor sizing with generalization.

V. Experiment Results

A. Experimental setup

Our experiments are conducted on both academic and industrial settings to evaluate the feasibility and the capability of the DRL agents.

During the development phase, the agents are tested on a two-stage opamp with BSIM 45/22nm processes simulated on an open sourced NGSPICE simulator developed by UC Berkeley. Two scenarios that are often encountered in practice, specifically, process porting and PVT exploration are analyzed.

To demonstrate that the proposed framework is beneficial in production, we cooperate with professional designers and evaluate the agents on two industrial cases with advanced TSMC technologies. Simulations are conducted in Cadence Spectre.

B. 45nm two-stage opamp

First, we benchmark our method with various baseline models including random search, customized BO (designed, implemented, and tuned by ourselves), Advantage Actor Critic (A2C)[26], Proximal Policy Optimization (PPO)[27], and Trust Region Policy Optimization (TRPO)[28] implemented by Stable-Baselines[29]. The customized BO utilizes dynamic balancing of exploration & exploitation, and also substitutes Gaussian Process with extra-tree regressor. All model-free agents follow the same observation design in AutoCkt, and use the same reward function as our agents. The experimental environment is set on a BSIM 45nm two-stage opamp simulated in a single PVT with a design space size of 10^{14}. The comparison is shown in Table .I.

Each experiment has a 10k-step limitation. For BO and our model-based agents, 100 experiments are ran to prove the stability, whereas for agents in stable-baselines, only 10 are executed since it would have taken about a month to complete.

TABLE I
Performance of agents in 45nm two-stage opamp

	Success rate	Average iterations
Random search	100%	8565
Customized BO	100%	330
A2C[26]	90%	34797
PPO[27]	40%	31503
TRPO[28]	20%	16350
Our method	100%	36

The experiments show that random search is a strong baseline in which model-free agents (A2C, PPO, TRPO) failed to reach its performance. The reason is that there exists a trade-off between gain and phase margin. Circuits with high gains often come with unstable phase margins. Hence, if the reward formula of model-free agents do not encode such information, the agent can be easily stuck in a local maxima where this happens.

BO states a solid stability among our experiments, however, the nature to estimate the global distribution gains iterations compared with our local model-based agents. As for model-based agents, all specifications can be met within 36 steps on average. Moreover, the standard deviation is 16 steps, which indicates that the search is stable. This comparison demonstrates that our idea of implementing an agent good at local exploration is feasible, and can outperform model-free agents by orders of magnitude.

C. Process porting

Many circuit topologies have to go through a process migrations. To avoid reinventing the wheels every time a new node is applied, IP reuse is an important topic worth discussing.

To better understand strategies for porting in our algorithm, a migration from 45nm to 22nm process is prepared. Each experiment is ran for 100 times to account for the randomness.

TABLE II
RESULTS OF PROCESS PORTING FROM 45NM TO 22NM

	Average steps	Min steps	Max Steps
Baseline (random weights, random starting points)[a]	50.17	15	191
Weight sharing, starting point sharing[b]	29.22	3	310
Random weights, starting point sharing[c]	20.74	2	88

[a]Directly deploy our method on 22nm without process porting
[b]Use the network weights and starting points when the model found an optimal solution in 45nm process
[c]Apply random weight initialization but start the agent on different optimal solutions found in 45nm process

All three experiments can 100% find solutions, so the success rate is omitted. Table. II shows that optimal points from previous nodes are reliable. However, the results state impotent network weights transferability, implying that the distributions between processes are distinctive. Interestingly, this phenomenon matches R&Ds' experiences: previously sized circuits are strong references, but the equations describing the physics of transistors could be distinguishing.

D. PVT exploration

Finding a sufficient set of sizes in a single condition is only a part of the story. Thus, we test PVT exploration strategies on our method using the two-stage opamp with 22nm process. The results are illustrated in Table. III.

TABLE III
COMPARISON OF PVT EXPLORATION STRATEGIES

	Average steps	Min steps	Max steps
Random search	failed (10,000+)	NA	NA
Brute force (test all cond.)	359.4	36	1305
Progressive (random cond.)	89.52	20	450
Progressive (hardest cond.)	72.60	15	279

Only random search cannot accomplish the task, other strategies can finish 100% of the time. A 4x improvement of progressive search over brute force search is exemplified. An intriguing finding is that while starting from the most difficult condition does make a difference, choosing a random corner to start also produce comparable results. This suggests that the progressive search is not sensitive to the initial condition, which is positive for cases where the toughest PVT corner is unidentifiable owing to the number of permutations.

E. Industrial case

LDO (Low-Drop regulator) on TSMC 6nm The first industrial example is a LDO with TSMC 6nm process. In this case, there are 4 transistors in the schematic, creating a design space of about 10^{29} possible combinations.

Since the circuit was under rapid development, the number of iterations of human designers is untraceable. Nevertheless, our agent achieved the specification in all 45 corners within 2609 iterations, which is considered fast in designers' opinion. Furthermore, the performance obtained surpass designers while producing even lower area, shown in table. IV. An interesting discovery is that even some of the device sizes decided by both AI and human are the same, AI still managed to illustrate an area enhancement.

In comparison with BO, it cannot satisfy all the constraints within a reasonable time. However, the best parameters searched is very close to the specifications.

TABLE IV
BENCHMARK OF LDO CIRCUIT SIZING WITH DIFFERENT AGENTS

	# iterations	Loop gain	Area
Human	untraceable	38.0dB	650
Customized BO[a]	failed	38.2dB	604
Our method	2609	40.0dB	632

[a]Customized BO did not satisfy the constraints (not shown here), however, it gives very close results

ICO (Current-Controlled oscillator) on TSMC 5nm An ICO is tested as the second case with TSMC 5nm process. The design space size is 20^4.

The AI-sized ICO realized a performance on a par with human designers. A comparable result is exhibited in BO. Yet, as mentioned before, the global search strategy cause 4.5 times more iterations than our local search algorithm (Table. V).

TABLE V
BENCHMARK OF ICO CIRCUIT SIZING WITH DIFFERENT AGENTS

	# iterations	Phase noise	FOM	Frequency
Human	untraceable	-73.31°	154.99	8.45G
Customized BO	194	-72.17°	154.22	8.87G
Our method	43	-71.76°	154.20	9.18G

Although the design space of the second case is relatively small, this leads to a declaration. Numerous concerns in the community are raised on AI safety[30]. As an initial assessment, the ability and the characteristics of the designed circuits are unfamiliar. Therefore, to ensure that the agents act as intended and to secure the safety of the products, designers have to fix a subset of the parameters, only letting the agents to search for the rest. To that end, not until we have a comprehensive rule for regulating the agents can we unlock the full capability of AI. Thus in our evaluations, designers went through a rigorous screening process to examine the designed circuits. Fortunately, the sized circuits are ready to tape out.

From both cases described earlier, the outcome is akin to what is addressed in SimPLe[17]. Transistor sizing is one of the task that benefits from the sample-efficient advantage of model-based agents.

In summary, the results state the contributions of this framework in two ways:

- provides better PPA within a reasonable time

- automates the process, leaving human intervention while obtaining granting satisfactory PPA

This also indicates that the model is generalizable across different circuit schematics and process nodes. The term generalization here does not refer to network weight level transfer as the convention in machine learning genre, but rather algorithm architecture level.

Discussion

This project has no means of replacing professional analog designers. In analog circuit design, especially when utilizing rapidly scaling process nodes, it is harder than ever for human designers to take account for second or even higher orders of effects on transistors. Even though some sections of the flow could be accomplished by AI agents, but AI would not be useful if the upstream tasks such as system-level architecture design and circuit-level topology selection were carefully executed by human experts with their rich experiences and knowledges.

The core value of this framework is to help increase R&D productivity, allowing designers to manipulate DRL agents to do what they are trained for specifically. Despite neural networks are hardly interpretable, human can still get inspiration from AI-produced creative actions, identical to the move 37 of AlphaGo. In this auxiliary fashion, designers still have to conduct a series of serious investigations into the quality of AI-designed circuits to make sure that the chip is production-ready.

In concern with the real capability of scalability, certainly, AI is not trained, nor designed to achieve full chip design since it is computationally infeasible. Rather we can embrace current limitations by divide-and-conquer. First feeding an appropriate amount of the design to our search framework, then abstract the block into a behavior model when all specifications are met. By recursion, we can avoid such scalability problem. Once again, it is the merit of designers to come up with a segmentation that allows to leverage the current AI technology.

Conclusions

In this work, we propose a generalizable search framework using learning-based algorithms for solving the analog transistor sizing problem. We take a novel direction where a local search strategy is adopted using model-based agents trained with supervised learning. This enables fast design space adaptation. Moreover, a PVT exploration strategy is also proposed to account for different working conditions.

Practical evaluations on industrial products with advanced TSMC 5/6nm process shows exceptional results. Our agent achieves performance beyond human level while producing smaller area. Furthermore, the presented framework is not limited to this specific stage of the flow. Any section that could be cast as a search problem can be transferred and leverage the assistant of DRL technology.

References

[1] M. Javaheripi, M. Samragh, and F. Koushanfar, "Peeking into the black box: A tutorial on automated design optimization and parameter search," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 4, pp. 23–28, 2019.

[2] K. Hakhamaneshi, N. Werblun, P. Abbeel, and V. Stojanović, "Analog circuit generator based on deep neural network enhanced combinatorial optimization," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–2.

[3] K. Wawryn, "An artificial intelligence approach to analog circuit design," *Journal of Circuits, Systems, and Computers*, vol. 1, no. 02, pp. 149–176, 1991.

[4] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Batch bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design," in *International Conference on Machine Learning*, 2018, pp. 3306–3314.

[5] S. Zhang, W. Lyu, F. Yang, C. Yan, D. Zhou, X. Zeng, and X. Hu, "An efficient multi-fidelity bayesian optimization approach for analog circuit synthesis," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–6.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[10] C. Tang, Z. Ye, and Y. Wang, "Parametric circuit optimization with reinforcement learning," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2018, pp. 197–202.

[11] J. P. S. Rosa, D. J. D. Guerra, N. C. G. Horta, R. M. F. Martins, and N. C. C. Lourenço, *Using Artificial Neural Networks for Analog Integrated Circuit Design Automation*. Springer International Publishing, 2020. [Online]. Available: https://doi.org/10.1007/978-3-030-35743-6

[12] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolic, "Autockt: Deep reinforcement learning of analog circuit designs," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 1–6.

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

[14] H. Wang, K. Wang, J. Yang, N. Sun, H. Lee, and S. Han, "Tts: Transferable transistor sizing with graph neural networks and reinforcement learning," in *ACM/IEEE 57th Design Automation Conference (DAC)*, vol. 2, 2020.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[17] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine *et al.*, "Model-based reinforcement learning for atari," *arXiv preprint arXiv:1903.00374*, 2019.

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

[19] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[20] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[21] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," *arXiv preprint arXiv:1611.02167*, 2016.

[22] S. Sadasivam, Z. Chen, J. Lee, and R. Jain, "Efficient reinforcement learning for automating human decision-making in soc design," in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.

[23] H. Zheng and A. Louri, "An energy-efficient network-on-chip design using reinforcement learning," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[24] D. Lian, Y. Zheng, Y. Xu, Y. Lu, L. Lin, P. Zhao, J. Huang, and S. Gao, "Towards fast adaptation of neural architectures with meta learning," in *International Conference on Learning Representations*, 2019.

[25] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," 2002.

[26] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

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

[28] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015, pp. 1889–1897.

[29] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[30] J. Leike, M. Martic, V. Krakovna, P. A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, and S. Legg, "Ai safety gridworlds," *arXiv preprint arXiv:1711.09883*, 2017.