RODESIGNER: VARIATION-AWARE OPTIMIZATION FOR ROBUST ANALOG DESIGN WITH MULTI-TASK RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Analog/mixed-signal circuit design is one of the most complex and time-consuming stages in the chip design process. Due to various process, voltage, and temperature (PVT) variations from chip manufacturing, analog circuits inevitably suffer from performance degradation. Although there has been plenty of work on automating analog circuit design under the typical variation condition, limited research has been done on exploring robust designs under the real and unpredictable silicon variations. To address these challenges, we present RoDesigner, a robust circuit design framework that involves the variation information in the optimization process. Specifically, circuit optimizations under different variations are considered as a set of tasks. Similarities among tasks are leveraged and competitions are alleviated to realize a sample-efficient multi-task training. Moreover, RoDesigner prunes the task space before multi-task training to reduce simulation cost. In this way, RoDesigner can rapidly produce a set of circuit parameters that satisfies diverse constraints (e.g., gain, bandwidth, noise...) across variations. We compare our method with bayesian optimization, evolutionary algorithm, and Deep Deterministic Policy Gradient (DDPG) and demonstrate that RoDesigner can significantly reduce required the optimization time by **14**×**-30**×. We also show that RoDesigner 's circuit performance is as good as a state-of-the-art human design, while the design time is reduced from several days by an expert to an hour.

1 INTRODUCTION

Analog circuit design is a paramount but extremely challenging task. It requires a huge amount of human efforts and lacks effective automations. Due to numerous chip manufacturing variations, analog circuits suffer from non-trivial performance degradation. Addressing such variation issues is considerably challenging. Large manufacturing variations make the circuit performance *unpredictable*. In the performance distribution visualized in Figure 1, quite a few proportion of chips are landing in the red regions that are completely defected and discarded. As the chip fabrication technology advances, variation issues become even worse, leading to a larger chip failure rate. If such severe variation issues are not carefully handled, significant economic losses up to the billions of dollars will occur (McConaghy et al., 2012). Hence, an effective variation-aware circuit design methodology is in high demand.

Traditional solutions to address such circuit variation issues primarily rely on laborious human expert involvement. Experts manually design the circuit based on *their expertise* and the feedback from *a large number of circuit simulations* and iterate the process until it passes all variation tests. However, the *burdensome analysis and slow simulations* make the manual design process considerably time-consuming.

Existing automated methods cannot address variation issues effectively. The black-box optimization algorithms (Cohen et al., 2015; Lyu et al., 2018) and learning-based automation techniques (Wang et al., 2020; Settaluri et al., 2020; Wang et al., 2018) are used to design circuits. However, they merely focus on the optimization under *the typical condition without variations*. None of them can systematically produce a robust design under *real chip variations*. The variation-aware optimization is challenging in two aspects. First, the *simulation cost is prohibitively expensive* in order to get accurate variation effects under many test cases. Second, different variation conditions might conflict

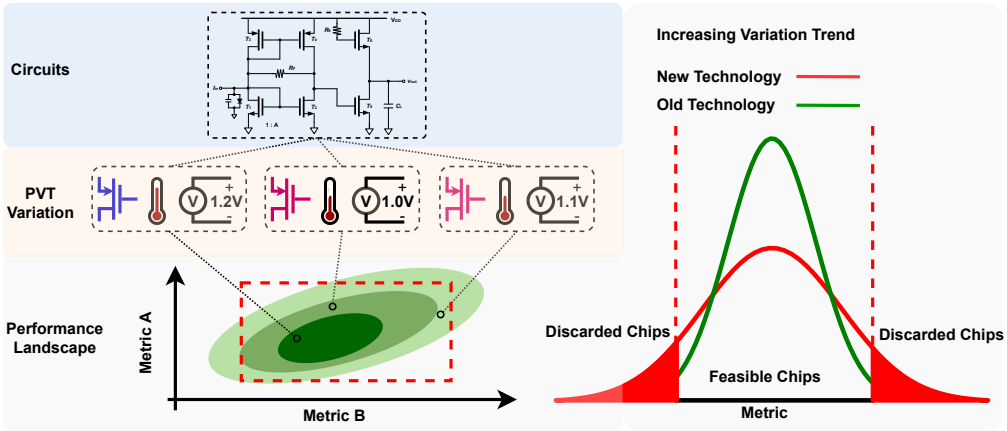

Figure 1: *Left:* Performances under variations form a distribution. *Right:* New technologies have larger process variations, vulnerability to environmental variations, hence higher discarded rate.

with each other which *significantly complicates the circuit optimization problem*. It will cost the solver much more time to find a feasible solution that meets all performance constraints.

To address the above challenges, in this work, we present RoDesigner, an efficient variation-aware optimization framework for automatic analog circuit design. RoDesigner largely reduces the simulation cost to design a robust analog circuit against variations. Here the variation-aware optimization is formulated as a multi-task reinforcement learning (RL) problem, where design for each variation condition is considered as one task. RoDesigner includes two stages. At the first stage, we select a representative subset of tasks as the training set. Specifically, we group the tasks using clustering algorithm and choose one task per group to form the training task set based on their relative performance to the target performance. At the second stage, we leverage multi-task deep deterministic policy gradient (DDPG) (Lillicrap et al., 2015) to train our RL agent with the selected tasks. During training, the critic model learns to predict values of state-action pair from each task and guides the actor to generate a better policy. To alleviate conflicting multi-task gradients, we apply PCGrad (Yu et al., 2020) to optimize actor and critic models.

The core contribution of this work is an optimization framework for addressing the variation issue efficiently. This variation-aware optimization framework comprises multi-task RL training and task-space pruning, bringing down the simulation cost. We conducted extensive experiments on three real-world circuits with practical variation requirements. Compared with evolutionary strategy (ES) (Cohen et al., 2015), bayesian optimization (BO) (Snoek et al., 2012), and DDPG methods, RoDesigner dramatically reduces the simulation cost needed to overcome variations. The simulation cost is cut down by **14**×**-30**×. The cost reduction becomes **4.4**× larger when scaling to the large task set. Moreover, RoDesigner's design solutions are comparable in performances to a state-of-the-art human design. While it takes several days for the expert to tune the circuit, RoDesigner can finish it within an hour. Our study enables the deployment of analog automation techniques on real silicon conditions.

2 Related Work

PVT Variation and Corners – The major part of variations is *PVT variation*. PVT variation usually refers to a combination of global process variation (P), power supply (V), and temperature (T) variations. Process variations happen during chip manufacturing, resulting in different transistor characteristics. There are five transistor models to cover the process variation {TT, SS, FF, SF, FS}. The T stands for typical, S for slow, and F for fast versions of transistors. Power supply voltage and temperature variations are in the end user environments. For example, circuits in space stations must handle extreme temperatures. To avoid circuit failures due to uncontrollable PVT variations, we model all these variations by a set of *PVT corners*. A PVT corner is a combination of process, voltage, and temperature values. For example, a fast-process, high-voltage, and low temperature corner is {Process = FF, V_{dd} = 1.3V, T = 15°C }. A robust circuit should maintain desired performances in all of the pre-set PVT corners.

Automatic Analog Sizing – An important stage of analog design is to size the devices in the given topology to get desire performances. Analog sizing is a challenging engineering design problem even without considering variations. There are two reasons. First, the correlation between the sizing vector and the multi-dimensional metrics is complex. Second, the sizing solutions are usually very sparse in the design space. The list of things that can go wrong is long. To deal with these challenges, designers usually reduce the design space to a small range based on manual analysis. Then designers rely on a large number of simulations to fine-tune the sizing parameters. Such a design process is labor-intensive and time-consuming. Therefore, automatic analog sizing techniques are attracting more and more research interests these years. The optimization methods, including Bayesian Optimization (Snoek et al., 2012; Lyu et al., 2018), Genetic Algorithms (Cohen et al., 2015) formulate the circuit design as a black-box problem. They show the differences in the sample efficiency and optimality. However, the critical issue is that they have to optimize the circuit from scratch every time when encountering a new design condition. The lack of tranferalibity across different conditions prevents them from addressing the variation issue at an affordable cost. Recently, learning-based methods have been extensively applied to circuit sizing problems. DRL methods show the potential to achieve higher circuit performances with enough explorations in the design space compared to the black-box optimization methods (Wang et al., 2018; 2020; Settaluri et al., 2020). Moreover, deep neural networks (DNN) (Li et al., 2019; Wang et al., 2020; Zhang et al., 2019) can approximate the complex relation between circuit parameters and performances. The learnt circuit representations and the RL formulations enable the transfer learning across different design conditions, including different technologies and pre/post-layout design stage. However, current methods cannot reach the design goal under different conditions simultaneously. In this paper, our work effectively addresses the variations of real-circuits altogether.

Multi-Task RL – Deep reinforcement learning (DRL) is an emerging subfield of RL that can scale RL algorithms to complex and rich environments. Recently, DRL has many successful applications such as robotics (Levine et al., 2016), AutoML (He et al., 2018), and chip floor-planning (Mirhoseini et al., 2021). Multi-task RL focuses on enabling the single agent to solve multiple related problems, either simultaneously or sequentially (Teh et al., 2017). Learning multiple related tasks together should facilitate the learning of each individual task (Bengio, 2012; Caruana, 1997; Taylor & Stone, 2011; Yosinski et al., 2014). But it has also been found that training on multiple tasks can negatively affect performance on each task. Different kinds of techniques are proposed to solve this issue including new architectures (Heess et al., 2016; Devin et al., 2017), auxiliary tasks (Jaderberg et al., 2016), and new optimization schemes (Hessel et al., 2019; Yu et al., 2020). Besides, choosing which task or tasks to train on at each time step is also important. The task scheduling (Sharma et al., 2017) is also discussed. The idea behind it is to assign task scheduling probabilities based on relative performance to a target level. Optimized training task selections can significantly improve model performance (Bengio et al., 2009). We explored both the optimal task selection and multi-task training. They are integrated together into the framework to boost the sampling efficiency.

3 Proposed PVT Variation-Aware Circuit Sizing

3.1 Problem Definition

Given a fixed circuit topology, we search for a circuit sizing vector whose performance can satisfy the constraints (design targets) across all variations. Then the problem can be formulated as a constraint satisfaction problem under different conditions.

$$\begin{aligned} &\text{minimize} \quad 0 \\ &\text{subject to} \quad F_i(X|T_j) < C_i, \quad j = 1, \ldots, k \end{aligned} \tag{1}$$

where

$$\begin{aligned} X &= X_1, X_2, ..., X_n \\ D &= D_1, D_2, ..., D_n \\ C &= C_1, C_2, ..., C_m \\ T &= T_1, T_2, ..., T_k \end{aligned}$$

X, the sizing vector, is an n-dimensional variable which corresponding to n circuit sizing parameters. D is the domain for X. For example, D_1 is [0, 1] which means the design space of X_1 is [0, 1]. T is the set of k pre-defined PVT corners to cover possible variations in the real world. C is the constraint set for all circuit metrics. Because we have m metrics, the number of constraints is also m. $F_i(X|T_j)$ is the i^{th} performance metric of circuit under the j^{th} corner. F_i is a non-linear mapping between X and the i^{th} metric in the performance. X is the input, and T is the parameter. We rely on the circuit simulator to provide this mapping. Therefore, our goal is to find an X that can satisfy any constraints in C under any corner task in T. It is worth noting that choosing which tasks to optimize is also non-trivial. Spending simulations on each task is wasteful and provides minimal additional information since the correlation among tasks is ignored. A more interesting way is to conduct the task selection and multi-task training jointly.

3.2 Framework Overview

An overview of the proposed framework is shown in Figure 2. We consider satisfying constraints under one PVT corner as a single task. In each iteration, (1) RoDesigner selects a new task subset from all PVT corner tasks. For the first iteration, a pre-defined nominal corner will be selected as the first task; (2) The RL agent generates actions and passes them to each environment in the task subset; (3) The environment denormalizes actions ([-1, 1] range) to actual circuit sizings and refines them. The sizings will be truncated according to minimum precision, lower and upper bounds of the technology node if necessary; (4)Simulate the circuit (5) Agent gets the rewards from corner-specific environments. Optimizations are performed on the actor and critic networks with PCGrad technique. (6) If all tasks in the subset are passed during the agent evaluation, the sizing solution will be tested on the full task set. If it passes all tasks, the loop terminates. Otherwise back to (1). In the meantime, actor-critic model weights and replay buffers are saved for the agent to inherit in the next iteration.

3.3 Automatic Sizing with Multi-Task RL

Multi-task RL is a training paradigm in which the agents are trained with samples from multiple tasks simultaneously. Shared representations are learnt from a collection of related tasks. These

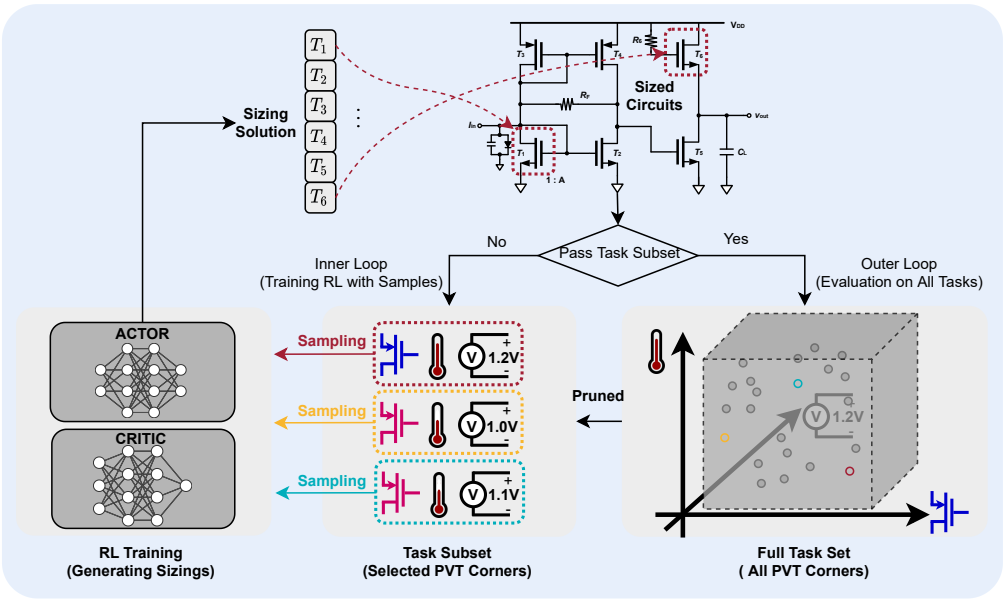

Figure 2: RoDesigner Overview. (1) A pruned task subset is generated from the full task set (2) Multi-task RL agent is trained on task subset (3) Training continues until the produced sizing can achieve training tasks. Then the sizing is evaluated on the full set. If it passes all the tasks, RoDesigner returns the result.

shared representations increase sample efficiency and can potentially yield a faster learning speed for related tasks. In our setting, we create a multi-task agent whose critic can predict the value of task-conditioned action-state pairs. Since the target of the actor is to look for a sizing that passes all tasks, the actor model is set to be task agnostic. Another benefit from shared representations is its ability to generalize to unseen corner tasks, which is useful in Monte Carlo corner tests. There are more discussions in Section 4.

State. The PVT information is embedded in our states, $s = (p, v, t)$, where p is the one-hot representation of component type, v is the normalized voltage value and t is the normalized temperature value.

Reward. Our reward is formulated as:

$$R = \begin{cases} r, & r < -0.02 \\ 0.2, & r \geq -0.02 \end{cases} \tag{2}$$

$$r = \sum_{i=1}^{M} \min\{\frac{m_i - m_i^*}{m_i + m_i^*}, 0\} \tag{3}$$

where m_i is the current simulated i^{th} performance metric and m_i^* is the corresponding constraint. The reward is a measure of the relative distance between the current performance metrics and the corresponding design targets. Once the requirements are met, the reward value is fixed at 0.2. This reward formulation is motivated by the design goal in the real world. Designers tend not to over-optimize the circuits. It is more important that designers can fulfill the requirements in a short period of time.

Action. The action vector is a set of values corresponding to the sizing parameters for each circuit. They include transistor sizes (width, length) and capacitor values. The details of settings for each benchmark are illustrated in section 4.

Training. The environment includes the circuit, simulator, and PVT information. Each time we query the environment, it simulates the circuit and returns the performance with PVT information. After agent-environment interactions, samples (s, a, r, z_i) will be stored in the replay buffers , where s is the state, a is the action, r is the reward, and z_i is the corner task ID. The critic neural network takes (s, a, z_i) as a input and predicts the corresponding value for the current corner task. Relying on the insight that performance under different corners are related, most of critic neural network parameters are shared across tasks except a few in the input layer. The task ID is removed from the inputs of the actor neural network. The training process is modified from DDPG (Lillicrap et al., 2015). Details are illustrated in Algorithm 1. M is the max optimization episodes and W is the warm-up episodes. N is the truncated norm noise. N_s is the training batch size. The key difference from the single-task setting is that we sample a stratified batch from buffers every time and generate task-specific losses. Also, samples from different tasks are stored in separated tasks. For the optimization strategy, we use PCGrad (Yu et al., 2020) to address conflicting gradients from different tasks.

3.4 Task Space Pruning

The straightforward ways to form a training task set include using the full task set and sampling tasks uniformly from the full set. Many multi-task algorithms use these two baseline methods to choose training tasks. In our case, the number of corner tasks is tied to the number of time-consuming simulations. This motivates us to create small-sized training tasks. Therefore, we apply k-means clustering (MacQueen et al., 1967) to prune the full task space. Given a circuit sizing, we simulate the circuit on all corner tests to get the corresponding performance distribution. Performances of each corner are high-dimensional vectors. We divide the corners into different groups by using the performances as the features. We select the corner with the lowest reward in each cluster as one of the training tasks in the next iteration. Relying on the insight that corners in the same cluster share similar learning dynamics, we only have to select one of them in our training tasks. By doing so, the task space for multi-task RL training in each iteration is pruned to be a significantly smaller scale while still being a good representation of the full task space. If there is no sizing given at first, a pre-defined nominal corner will be chosen as the first corner to train on. An interesting finding from

Algorithm 1: Proposed Automatic Sizing with Multi-task RL

Given critic network $Q(S, A, Z_i \mid \theta^Q)$ and actor network $\mu(S \mid \theta^\mu)$ with critic weights θ^Q and actor weights θ^μ ;
Given replay buffers $\{P_i\}$;
for *episode = 1, M* **do**
  Initialize random process $\mathcal{N}$; Reset all environments S;
  if *episode* $\leq W$ **then**
    Warm-up: randomly sample an action $\widetilde{A}$;
  else
    Select action $\widetilde{A} = \mu(S \mid \theta^\mu) + \mathcal{N}$ according to the current policy and exploration noise;
  Denormalize and refine $\widetilde{A}$ with design constrains to get A;
  Simulate the A for each task to get rewards $\{R_i\}$;
  Store each transition (S, A, R, Z_i) in P_i;
  if *episode* $> W$ **then**
    Sample a stratified batch of $(\widehat{S}, \widehat{A}, \widehat{R}, \widehat{Z_i})$ from $\{P_i\}$ (batch size = N_s);
    Update the critic by minimizing K losses with PCGrad:
    $L_i = \frac{1}{N_s} \sum_{k=1}^{N_s} (\widehat{R}_k - B - Q(\widehat{S}_k, \widehat{A}_k, \widehat{Z_i} \mid \theta^Q))^2$;
    Update the actor using the K gradients modified by PCGrad:
    $\nabla_{\theta^\mu} J_i = PCGrad(\frac{1}{N_s} \sum_{k=1}^{N_s} \nabla_a Q(S, A, Z_i | \theta^Q)|_{\widehat{S}_k, \mu(\widehat{S}_k)} \nabla_{\theta^\mu} \mu(S|\theta^\mu)|_{\widehat{S}_k})$;

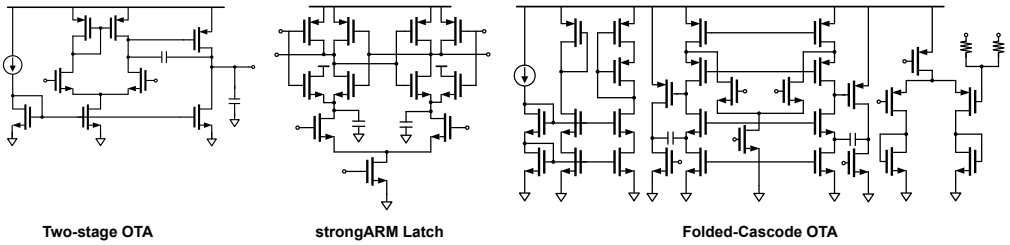

Figure 3: Three analog/mixed-signal benchmarks.

the empirical study results is that the easier corner tasks help to accelerate the learning of other hard tasks. Therefore, we always add a nominal corner as an auxiliary task in the training task sets at all time steps. If all the corners are passed, the loop terminates.

4 Experiments

4.1 Analog/Mixed-signal Circuits

We experiment with three real-world analog/mixed-signal circuits. They are two-stage operational transimpedance amplifier (Two-stage OTA), folded-cascode operational transimpedance amplifier (Folded-Cascode OTA) and strongARM Latch. They are chosen for three reasons. First, they are the most important and common-used blocks in various systems. Engineers usually spend the longest time optimizing the performance and robustness of these circuits. Second, they include two representative kinds of analog circuits which are the static and dynamic circuits. The two kinds are dictated by different physic and engineering rules. The third reason is that they have different levels of variations. Two-stage OTA is with 45nm, and the other two are with older 180nm technology. 45nm has a larger variation. Therefore, we can study the impacts of different variation magnitudes. Each circuit is a composition of a number of transistors and capacitors. Each transistor has two parameters, the gate width and length (w, l). Capacitors have one parameter (c), the capacitance value. The initial design spaces of these devices are given by human designers. To minimize the efforts of designers, our design space are set to be very large. They have 10^{14}, 10^{27}, and 6.4×10^{64} possible values correspondingly. The circuits are simulated on SPICE-based simulators (Nagel & Pederson, 1973). Two-stage OTA is on Ngspice and BSIM 45nm predictive technology. Folded-Cascode OTA

and strongARM Latch are on Cadence spectre and TSMC 180nm technology, a commercial simulator tool. For strongARM Latch, we have a published, state-of-the-art human design (Tang et al., 2020) which can be compared with the solution produced by RoDesigner.

Two-stage OTA. The topology is shown in Figure 3. It has 7 parameters including 6 transistor widths (w) and 1 capacitor value (c). The range of w is [0.5, 50]$*1\mu M$ and [0.1, 10]$*1pF$ for c. The total design space is 10^{14} possible values. The performance metrics are current(i), unity gain-bandwidth (ugb), phase margin (phm). The corresponding constraints (C) and the PVT corner tests (T) are showed below. There are 30 corners $(5 \times 3 \times 2)$.

$$T = \{TT,\ SS,\ FF,\ FS,\ SF\} \times \{1.0V,\ 1.1V,\ 1.2V\ \} \times \{0°C,\ 100°C\ \}$$
$$C = \{i \leq 5mA,\ ugb \geq 15MHz,\ phm \geq 60°\}$$

Folded-Cascode OTA. The topology is shown in Figure 3. It has 20 parameters, including 7 transistor widths (w), 7 lengths (l), 2 capcitor values (c) and 4 transistor ratios (n). The range of w is [0.24, 150]$*1\mu M$, [0.18, 2]$*1\mu M$ for l, [0.1, 2]$*1pF$, [0.1, 10]$*pF$ for different c. The total design space is 6.4×10^{64} possible values. The performance metrics are power(p), unity gain (g), phase margin (phm), common-mode rejection ratio (CMRR), power supply rejection ratio (PSRR), noise (n), unity-gain-bandwidth (ugb). The corresponding constraints (C) and the PVT corner tests (T) are showed below. There are 20 corners $(5 \times 2 \times 2)$.

$$T = \{TT,\ SS,\ FF,\ FS,\ SF\} \times \{1.6V,\ 1.8V\ \} \times \{0°C,\ 100°C\ \}$$
$$C = \{p \leq 1mW,\ ugb \geq 30MHz,\ phm \geq 60°,\ n \leq 30mV,\ g \geq 60dB,\ \}$$

strongARM Latch. The topology is shown in Figure 3. It has 7 parameters, including 6 transistor widths (w), 1 capcitor values (c). The range of w is [0.22, 50]$*1\mu M$, [0.15, 4.5]$*1pF$ for c. The total design space is 10^{27} possible values. The performance metrics are power(p), set delay (sd), reset delay (rd), set voltage (sv), reset voltage (rv), noise (n). The corresponding constraints (C) and the PVT corner tests (T) are showed below. There are 20 corners $(5 \times 2 \times 2)$.

$$T = \{TT,\ SS,\ FF,\ FS,\ SF\} \times \{1.1V,\ 1.2V\ \} \times \{0°C,\ 100°C\ \}$$
$$C = \{p \leq 4.5uW,\ n \leq 50uV,\ sd \leq 14ns,\ rd \leq 9.1ns,\ sv \geq vdd - 0.05V,\ rv \leq 0.05V\ \}$$

4.2 Training Settings

To demonstrate the effectiveness of the proposed RoDesigner, we apply RoDesigner to the above three circuits and record the simulation time it took to pass all the corner tests. We compare the results of RoDesigner with Bayesian Optimization (BO) (Snoek et al., 2012), Evolutionary Strategy (ES) (Hansen, 2016), and Deep Deterministic Policy Gradient (DDPG). For the three baselines, the variation-aware circuit optimization is considered as a single task. The average reward of all corner tasks is used to indicate the goodness of the current sizing. BO, ES, and DDPG improve the average reward until it reaches 0.2. In ES, DDPG, and RoDesigner, the circuit simulation time accounts for over 95% of the total time. The computation time of BO becomes comparable with simulation time after many iterations. We compare these methods in terms of the simulation time. For RL training, we use a training batch size of 64, replay buffer size of 1000, and exploration noise standard deviation of 0.2. Actor and critic are all 4-layer multilayer perceptions (MLPs). For RL methods, we evaluate the agent every 10 training steps. All the experiments are conducted on a 6 core CPU. RL methods are implemented with PyTorch (Paszke et al., 2019; Stooke & Abbeel, 2019)

4.3 Evaluation of the Circuit Optimization

In all three circuit benchmarks, RoDesigner achieved the smallest simulation cost to accomplish all the corner tasks. In each benchmark, it passed all the corners in the runs of different random seeds hence a 100% success rate. The comparison of simulation costs are shown in Figure 4. RoDesigner consistently outperforms the baseline methods including ES, BO, and single-task DDPG. The simulation cost reductions are huge, 26x in Two-Stage OTA, 30x in strongARM Latch, and 14x in

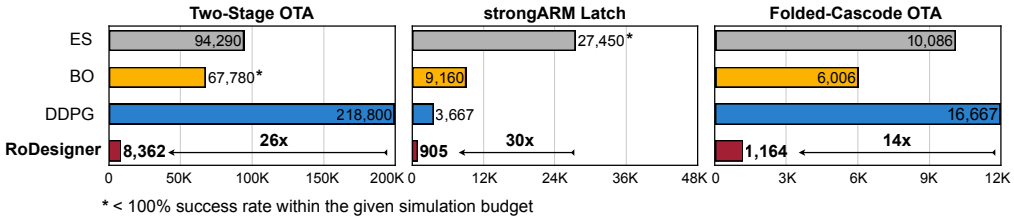

Figure 4: Simulation times for each method to take to first hit reward=0.2

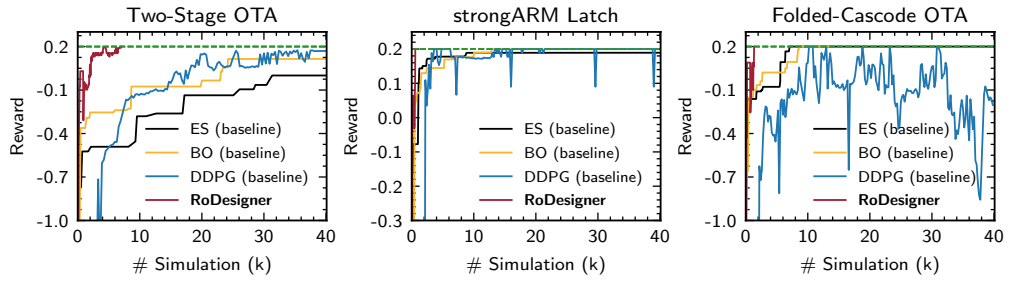

Figure 5: Compare learning curves (average reward vs. # simulation) among baselines and our proposed RoDesigner. Reward=0.2 indicates all tasks are passed. RoDesigner hits the reward of 0.2 significantly faster than the baseline methods on all benchmarks.

Folded-Cascode OTA. Note that BO becomes slow after having many samples. We ran BO for the same time with other methods for fair comparisons. We have several findings from the experiment results. First, all methods spend more simulations on optimizing the Two-Stage OTA which has larger variations with the 45nm technology. Second, compared to the ES and BO, single-task DDPG performs better in strongARM Latch while worse in the Two-Stage and Folded-Cascode OTAs. This is possibly because strongARM Latch is a dynamic circuit that is different from the static OTAs. To conclude, RoDesigner shows a significant efficiency improvement in the different levels of variations and circuit benchmarks with distinct natures. The learning curves are shown in Figure 5.

4.4 Analysis

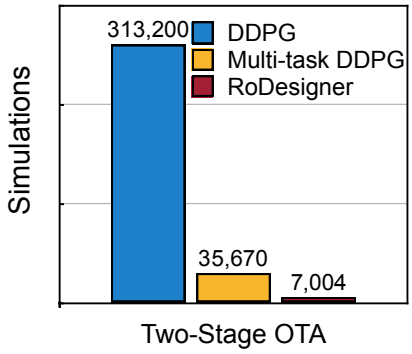

Figure 6: Ablation of applying multi-task and task space pruning. Using two together brings the least simulation cost

Multi-Task and Task Space Pruning. We conduct an ablation study on multi-task and task space pruning. In Figure 6, We compared simulation costs of DDPG baseline, multi-task DDPG with full task set, and RoDesigner (multi-task DDPG with pruned task set). DDPG took over 300,000 simulations to pass all corner tests. With the multi-task training, the number of simulations was reduced to 35,000. With the pruned task space, the number of simulations was further cut down to 7,000. We also visualize the corner performances and the optimization trace in the performance plane of three circuit benchmarks in Figure 7. Noise (n) - Delay (sd) plane is chosen for strongARM Latch and Bandwidth (ugb) - Phase Margin (phm) for OTAs. Selected training tasks are denoted by black circles. We can clearly see that selections are located at the performance boundary. Two snapshots of the performance distribution during the optimization are also showed. They clearly indicate that distributions moved towards the feasible set area from t_0 to t_1 with such pruned task space.

RoDesigner vs. Human Expert. To examine the quality of the solutions from RoDesigner, we compared them with a state-of-the-art human design. The performance metrics are listed in Table 1. Each metric is shown in the format of (min, max) across corners. For all metrics, RoDesigner

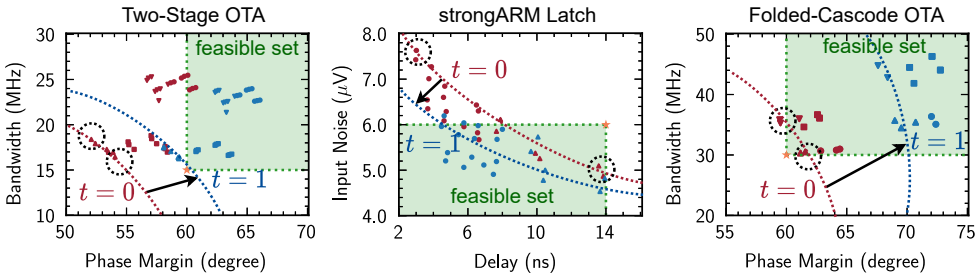

Figure 7: Performance distributions of two intermediate sizings during the RoDesigner optimization. Red and blue markers are performances on different corners at time t_0 and t_1. Selected training corners are indicated by black circles.

Table 1: Comparison between RoDesigner's solution and expert's solution

	Power (uV)	Set Delay (ns)	Reset Delay (ns)	Noise (uV)	DSV (V)	Reset Voltage (nV)
Human Expert Tang et al. (2020)	(3.78, 4.69)	(6.42, 19.7)	(5.02, 9.40)	(41.9, 57.3)	(0, 0)	(8.71, 1.99k)
RoDesigner	(2.22, 2.88)	(4.46, 13.9)	(1.30, 2.3)	(45.3, 61.9)	(0, 0)	(20.4, 2.21k)

performed better excepts for the slightly inferior noise performance. This benchmark, strongARM Latch, has non-linear behaviors and variation-sensitive performances. A large amount of tuning efforts is required. It can take days for the expert to achieve the design target. Now RoDesigner can achieve the same task and produce high-quality solutions within an hour.

Scale to Large Corner Sets. Here we empirically study how the simulation cost scales as we take on more and more corner tasks. In the previous sections, we discussed the fully factorial corner test for each benchmark. In industry-level circuits, randomly sampled corners, Monte Carlo corners, are also used. There can be hundreds, even thousands of Monte Carlo corners needed to perform a thorough verification. Therefore, the scalability to a large corner set is important. To demonstrate the scalability of RoDesigner, we conduct Monte Carlo sampling on process variation modelsets {TT, FF, SS, FS, SF}, continuous voltage range [1.0, 1.2] and continuous temperature range [0°C, 100°C] and form 5 Monte Carlo corner test sets of different sizes. These Monte Carlo corner sets have 20, 40, 80, 100, 150 corners, respectively. Experiments are done on Two-Stage OTA benchmark and results are shown in Figure 8. RoDesigner only needs 69% more simulations when the corner task set becomes 7.5× larger. The simulation cost difference between RoDesigner and the baseline methods will become 4.4× larger at the scale of 150 corners.

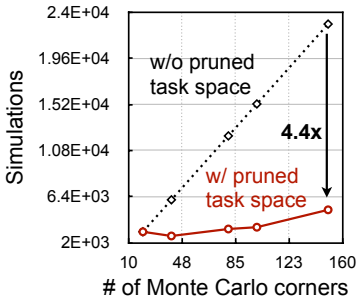

Figure 8: Required simulation steps with more corners

5 Conclusion

We present RoDesigner, a variation-aware optimization framework based on multi-task RL. The key property of RoDesigner is the ability to conduct efficient multi-task learning with pruned training task space. Therefore, it can effectively design circuits for variations. We show that RoDesigner can reduce simulation cost by an order of magnitude compared with baselines. It can also scale to a large number of variation cases. We also show that RoDesigner's solution is superior to the state-of-the-art human design in a popular circuit block. As today's chip design becomes extremely challenging with the presence of variations, RoDesigner shows the potential to drastically shorten the circuit design cycle and reduce the cost.

ETHICS STATEMENT

We do not find insights, methodologies of this work potentially harmful to ethnicity. Automating the circuit design is important because it can shorten the design cycle. As today's technology becomes more advanced and our computational needs becomes more diverse, the design complexity grows exponentially. The effective automation techniques can address this bottleneck and accelerate the semiconductor development.

REPRODUCIBILITY STATEMENT

Since the optimization and RL methods have stochasticity, we ran each of methods with three different random seeds. The anonymous code has been put into the supplementary materials. Note the confidential information of circuit technology has been removed from the materials.

REFERENCES

Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 17–36. JMLR Workshop and Conference Proceedings, 2012.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Miri Weiss Cohen, Michael Aga, and Tomer Weinberg. Genetic algorithm software system for analog circuit design. *Procedia CIRP*, 36:17–22, 2015.

Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2169–2176. IEEE, 2017.

Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Yihui He et al. AMC: AutoML for Model Compression and Acceleration on Mobile Devices. In *ECCV*, 2018.

Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3796–3803, 2019.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Sergey Levine et al. End-to-end training of deep visuomotor policies. *JMLR*, 2016.

Yaping Li, Yong Wang, Yusong Li, Ranran Zhou, and Zhaojun Lin. An artificial neural network assisted optimization system for analog design space exploration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(10):2640–2653, 2019.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Wenlong Lyu, Fan Yang, Changhao Yan, Dian Zhou, and Xuan Zeng. Batch bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design. In *International conference on machine learning*, pp. 3306–3314. PMLR, 2018.

James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pp. 281–297. Oakland, CA, USA, 1967.

Trent McConaghy, Kristopher Breen, Jeffrey Dyck, and Amit Gupta. *Variation-aware design of custom integrated circuits: a hands-on field guide*. Springer Science & Business Media, 2012.

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

Laurence W. Nagel and D.O. Pederson. Spice (simulation program with integrated circuit emphasis). Technical Report UCB/ERL M382, EECS Department, University of California, Berkeley, Apr 1973. URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/1973/22871.html.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

Keertana Settaluri, Ameer Haj-Ali, Qijing Huang, Kourosh Hakhamaneshi, and Borivoje Nikolic. Autockt: Deep reinforcement learning of analog circuit designs. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 490–495. IEEE, 2020.

Sahil Sharma, Ashutosh Jha, Parikshit Hegde, and Balaraman Ravindran. Learning to multi-task by active sampling. *arXiv preprint arXiv:1702.06053*, 2017.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *arXiv preprint arXiv:1206.2944*, 2012.

Adam Stooke and Pieter Abbeel. rlpyt: A research code base for deep reinforcement learning in pytorch, 2019.

Xiyuan Tang, Linxiao Shen, Begum Kasap, Xiangxing Yang, Wei Shi, Abhishek Mukherjee, David Z Pan, and Nan Sun. An energy-efficient comparator with dynamic floating inverter amplifier. *IEEE Journal of Solid-State Circuits*, 55(4):1011–1022, 2020.

Matthew E Taylor and Peter Stone. An introduction to intertask transfer for reinforcement learning. *Ai Magazine*, 32(1):15–15, 2011.

Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning, 2017.

Hanrui Wang, Jiacheng Yang, Hae-Seung Lee, and Song Han. Learning to design circuits. In *NeurIPS Machine Learning for Systems Workshop*, 2018.

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

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*, 2014.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *arXiv preprint arXiv:2001.06782*, 2020.

Guo Zhang, Hao He, and Dina Katabi. Circuit-GNN: Graph neural networks for distributed circuit design. In *International Conference on Machine Learning*, pp. 7364–7373. PMLR, 2019.

A APPENDIX

A.1 IMPROVE SAMPLE EFFICIENCY WITH PERFORMANCE PREDICTOR

To further improve the sample efficiency and reduce the simulation times, we propose a supervised learning method to obtain the performance of multiple corners only with one time of simulation. Specifically, for each step during the reinforcement learning, we only use the circuit simulator to simulate one corner, and the performance of other corners is predicted with a performance predictor.

We experimented with the two-stage OTA circuit. We collect a dataset of 10,000 random samples. Each sample contains the sizing and performance on all the corners. The predictor takes seven sizings, and four performance metrics on TT (temperature 27.0 and VDD 1.2v) as inputs and directly regresses the performance metrics of all other 30 corners (120 in dimension). We leverage a six-layer multi-layer perceptron model with hidden dimension 512. The 10,000 samples are split to train:valid:test=8:1:1. We use Mean Square Error (MSE) loss, Adam optimizer with 5.0e-4 learning rate, and weight decay lambda as 1.0e-4. The model is trained for 100 epochs with batch size 64. It can provide accurate performance estimations with RMSE of around 0.08.

Since the predictor is trained with randomly generated data, it is challenging to provide estimations for high-performance sizings. The predicted rewards of PVT corner tasks guide the agent to approaching the desired solution. But we still need real simulations to reach the goal ultimately. Therefore, we only employ the predictor in the early stage (first two task subsets) of the optimization. Table 2 shows the number of simulations with or without predictor on the two-stage OTA circuit. We run twice with different seeds. In the first run, the predictor reduces simulation times by 7,130; nevertheless, the predictor cannot reduce simulation in the second run. One future step to improve this is to use the dataset generated with RL or evolutionary search so that the predictor can provide accurate results for data points near the final goals.

Table 2: Number of Simulations Using Performance Predictor

	#sim w/o predictor	#sim w/ predictor	#sim difference
Run 1	20,839	13,529	-7,130
Run 2	5,049	9,694	+4,645

A.2 RODESIGNER LIBRARY

To optimize circuit by reinforcement learning (RL) and other algorithms, we built RoDesigner Framework. Each circuit benchmark is wrapped up as an environment including the circuit netlist, PDK, and the simulator. For RL part, we leverage the open-source library, rlpyt. In summary, RoDesigner provides the following features: (1) It supports different netlist, PDK, and simulators. Design targets and testbench can be easily configured by circuit designers by choosing their familiar tools. (2) It supports parallel environment query. Multiple circuit environments with different PVT settings can be run simultaneously. Also, rlpyt provides parallel workers which are multiple agent-environment pairs. These parallelisms accelerates the optimization process since simulation time is dominant in the process. (3) It provides detailed circuit metrics logging for debugging. In the meantime, rlpyt provides learning diagnostics logging to debug RL dynamics.

We include the code as a `.zip` file within supplementary materials.