The Dissertation Committee for Ahmet Faruk Budak
certifies that this is the approved version of the following dissertation:

Efficient Optimization Methods for Analog/Mixed-Signal Integrated Circuits via Machine Learning

Committee:

David Zhigang Pan, Supervisor

Michael Orshansky

Nan Sun

Yaoyao Jia

David Smart

Efficient Optimization Methods for Analog/Mixed-Signal Integrated Circuits via Machine Learning

by

Ahmet Faruk Budak

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2023

Dedication

Dedicated to my wife and to my parents.

Acknowledgments

I want to express my deepest thanks to my advisor, Dr. David Z. Pan, who has guided me through my graduate studies. I am utterly grateful for his vision and his mentoring. He is the shareholder of all my academic development, and I will always look up to his academic devotion, persistence, and organizational skills.

I would also like to thank my committee members: Dr. Michael Orshansky, Dr. Nan Sun, Dr. Yaoyao Jia, and Dr. David Smart, for their valuable advice and discussions. Specifically, I extend my most profound appreciation to Dr. Nan Sun. He has been a great co-advisor to my studies for several years and has always given me excellent technical and moral support. He also contributed to my multi-disciplinary research skills by helping to understand analog design automation problem from a designer's perspective. I sincerely thank Dr. Michael Orshansky for all his valuable feedback and for foreseeing my academic progress. I express my gratitude to Dr. Yaoyao Jia for numerous active discussions. I learned a great deal about analog design issues during the time when I collaborated with her and her research group. Also, I thank Dr. David Smart, whose one-on-one technical discussions I always enjoyed with. In many cases, he directly improved my research through his

constructive criticism and technical approach.

It has been a great honor to meet and work with all the UTDA lab members: Dr. Yibo Lin, Dr. Derong Liu, Dr. Biying Xu, Dr. Jaydeep Mitra, Dr. Wuxi Li, Dr. Shounak Dhar, Dr. Zheng Zhao, Dr. Wei Ye, Dr. Mohamed Baker Alawieh, Dr. Wei Shi, Dr. Miguel Gandara, Dr. Shaolan Li, Dr. Xiyuan Tang, Dr. Shuhan Zhang, Dr. Keren Zhu, Dr. Mingjie Liu, Dr. Hao Chen, Dr. Chenghao Feng, Dr. Jiaqi Gu, Rachel S. Rajarathnam, Zixuan Jiang, Hanqing Zhu, Xuyang Jin, Hyunsu Chae, Zhili Xiong, Chen-Hao Hsu, and Souradip Poddar. I enjoyed working with all of them and made great friends. I also thank the Nan Sun Research Group members who have contributed my research by their expertise in analog design. I especially thank Dr. Miguel Francisco Gandara, Dr. Wei Shi, Dr. Linxiao Shen, Dr. Shaolan Li, Dr. Xiyuan Tang, and Dr. Wenda Zhao for their active help through sharing their expertise in analog circuit design.

I want to thank Dr. Bo Liu, who introduced me to analog design automation field and helped me overcome many barriers in my first years. I want to thank my industry mentors and co-authors, Dr. Brian Swahn from Analog Devices Inc. and Dr. Chandra Kashyap from Intel Corp., for their technical support and valuable discussions. I also thank Prateek Bhansali for his research contributions and his mentoring during my internship at Intel Corp. Also, I cannot leave UT without acknowledging the UT ECE staff,

especially Melanie Gulick and Andrew Kieschnick, for their continuous support in the administrative and IT issues, which smooth the path for my research. I am also very appreciative of my other co-authors, teachers, and friends who helped in my study or research, too.

Finally, I am grateful for my family and for my wife, who are the true wealth I have in this life. My mother, Hamiyet, and my father, Ibrahim, have always shown their unconditional love and offered their support by any means possible. My wife, Elif, is the greatest person in my life and is the ultimate motivation source to survive any challenge. I hope anything I learned, and any progress I made will help me become a better person for those wonderful people.

Abstract

Efficient Optimization Methods for Analog/Mixed-Signal Integrated Circuits via Machine Learning

Ahmet Faruk Budak, PhD
The University of Texas at Austin, 2023

SUPERVISOR: David Zhigang Pan

During the analog design process, a significant amount of human effort is spent on optimizing circuit specifications by tuning the device parameters. Sizing device parameters is the task of obtaining satisfactory performance for certain constraint metrics and minimizing/maximizing other objective metrics. In general, an initial optimization is conducted based on schematic-level electrical simulations. However, the Analog/Mixed-Signal (AMS) Integrated Circuits (IC) design is also sensitive to the parasitics introduced during the layout. Therefore, a more comprehensive approach is to size device parameters under the consideration of layout parasitics. To automate this process, many automation methods are proposed where simulation feedback is integrated into

the automation loop for an accurate evaluation of design choices. AMS simulations are typically costly to run; therefore, the automation method's cost is crucial. This dissertation proposes efficient automated solutions to solve the AMS sizing problem.

First, this dissertation proposes a novel Machine Learning (ML) assisted evolutionary algorithm to tackle analog sizing problem. We address the data scarcity issue by introducing a data augmentation method that facilitates and improves the modeling of design metrics via Artificial Neural Networks (ANN). Further, we borrow techniques developed for evolutionary algorithms and introduce a parameter-free ranking methodology to differentiate design performance without human input. We assess the performance of our approach on several academic circuits and show that ML-based modeling significantly improves the simulation cost of the optimization algorithm.

Second, in this dissertation, we study applying Reinforcement Learning (RL) to solve analog sizing problem. We are influenced by the state-of-the-art policy gradient methods and tailor them to solve analog sizing task. Further, we include a recipe to extend this method for solving industrial-scale circuits with thousands of devices. We demonstrate the performance of our approach both on academic circuits and industrial circuits. We observe a significant performance improvement compared to several conventional baseline algorithms and compared to existing commercial tools.

Then we visit the AMS tasks with varying simulations costs. Motivated by the fact that one typically needs to run multiple types of simulations, we leverage cheap-to-run simulations to make intermediate decisions on the potential quality of explored points. Then we refrain from expensive-to-run simulations if necessary. In addition, we introduce an asynchronously parallel framework and adapt our previous work for the case of designs with the differentiated cost of simulations. Our benchmarking shows that the proposed methods significantly reduce the total real-time optimization cost and the total CPU effort.

Finally, this dissertation includes a solution on how to solve the sizing problem under layout effects effectively. We conduct a study to quantify the impacts of considering layout during transistor sizing. Then, we apply a Bayesian Neural Network (BNN) based approach to solve the sizing problem. To include layout-induced parasitics, we extend our approach via Multi-Fidelity BNN, where the algorithm utilizes multiple information sources for efficient learning of post-layout performances. We also include a search-space exploration strategy using the trust-region approach, which is shown to be effective on problems with high number of input dimensions. Our tests suggest that the BNN-based sizing algorithm is very competitive compared to previous state-of-the-art algorithms. We further demonstrate that the co-learning strategy of Multi-Fidelity BNN further improves the efficiency, which is very

crucial considering the costly post-layout simulations.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Analog/Mixed-Signal (AMS) integrated circuits (IC) are crucial elements of modern electrical systems. Figure 1.1 demonstrates that the global IC industry is a huge market with increasing demand, and analog has a considerable share. Especially the emerging Internet of Things (IoT) applications, autonomous and electric vehicles, communication, and 5G networks are the driving areas for analog demand. The increasing demand imposes tightened time-to-market requirements both on analog and digital circuits. Therefore, utilizing more automated solutions during the analog design process is a candidate solution to mitigate increasing market demand and shortened production cycle.

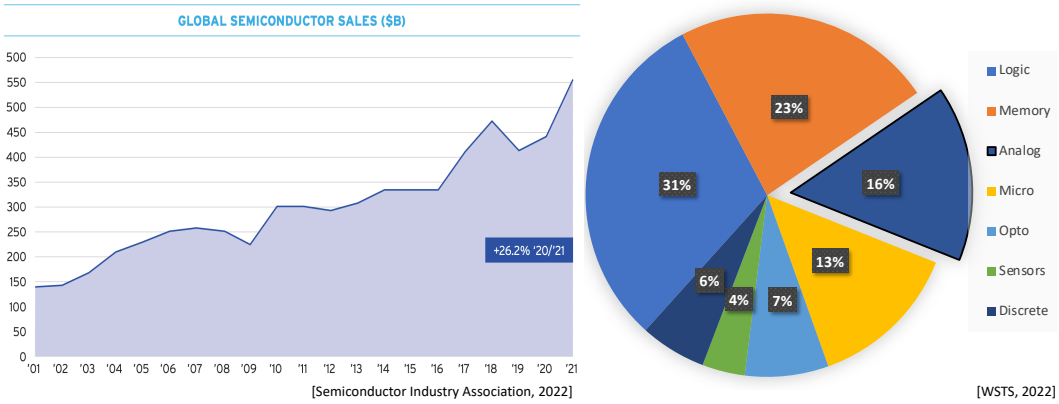


Figure 1.1: Global IC Market Value and Share of ICs by Type

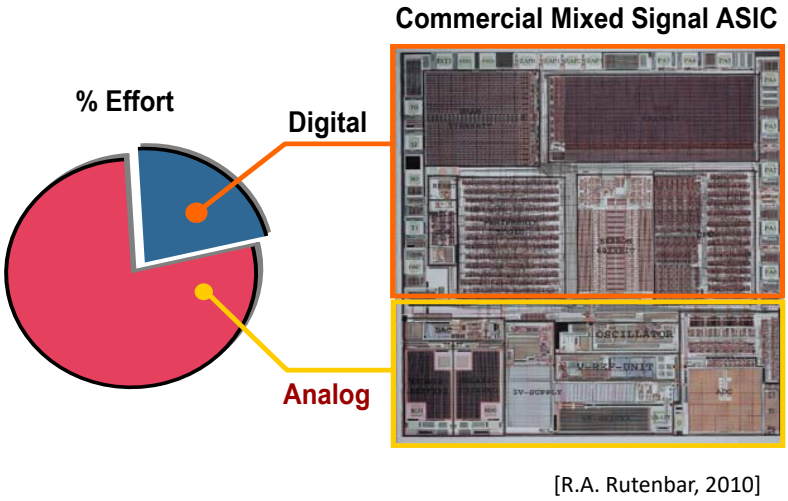


Figure 1.2: Design Effort of Analog vs. Digital

On top of the market demand for analog automation, there are additional motivations for improving analog automation. One is that the current analog design practice has considerable repetitions. Sometimes one design is ported from one technology node to another, or the same design may need to be reconfigured for a different performance specification. Another motivation is that although the digital takes the larger physical portion of a system, the analog portion suffers from a larger design effort (Figure 1.2). However, AMS blocks such as sensors, antennas, transmitters, and reference circuits are very essential and can not be replaced by digital. Therefore, analog is here to stay, and automating its design is a great academic and industrial interest.

The rest of this chapter is organized as follows. Section 1.1 introduces

AMS sizing automation and specifies the problem under different evaluation methods. We review schematic-level and layout-aware sizing methods in this section. Then, section 1.2 includes the overview of this dissertation.

1.1 AMS Sizing Automation Problem

Analog/Mixed-signal (AMS) integrated circuit (IC) design typically follows a process flow visualized in Figure 1.3. A combination of designer experience and computer simulation feedback is iterated to determine the design that meets the performance requirements. A large portion of design time is spent on the sizing and layout phases, where multiple iterations are possible due to potential loop-backs in the design flow. This is a labor-intensive process in industry practice with little to no automation. To address this costly exercise, a considerable effort in academia and industry is focused on introducing automated solutions.

Analog sizing automation is the task of optimizing AMS design variables, e.g., transistor widths, lengths, resistor, and capacitor values. The aim is to satisfy the performance constraints and optimize the design objective. In general, sizing automation is run through schematic-level simulations. However, AMS IC performance is also sensitive to layout implementation (10). Especially in the advanced process nodes, layout-induced parasitics may significantly affect the final design performance. Therefore, sizing the AMS design

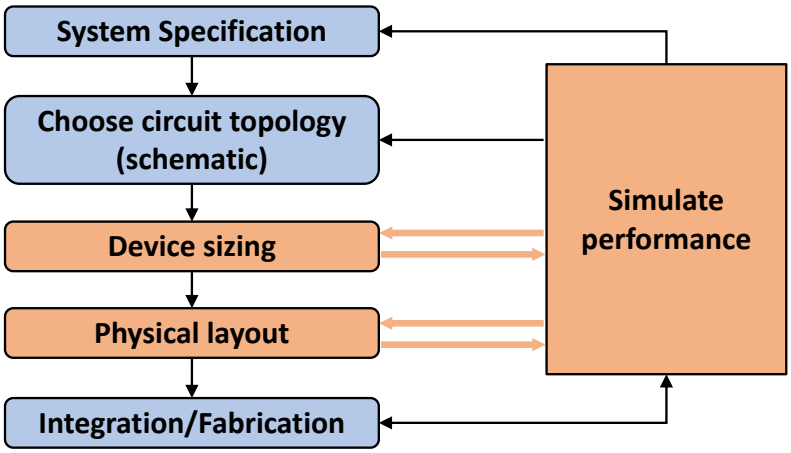


Figure 1.3: AMS Design Flow

variables considering the layout effects is also crucial.

The majority of the recent sizing and post-layout performance optimization algorithms have simulation feedback in the loop. Due to advanced scaling, simulations are required to obtain accurate performance evaluations. Simulation-based AMS automation algorithms adapted various methods from the optimization and Machine Learning (ML) communities. The earlier approaches include population-based methods such as particle swarm optimization (85) and evolutionary algorithms (46). Although these algorithms have good convergence behavior, they are inefficient in sampling since they explore the design space randomly. To mitigate sample inefficiency, model-based methods gained popularity (41; 57; 29). These methods employ surrogate models between the solution space and performance space and provide efficiency in

exploring the solution space. A typical surrogate model is Gaussian Process Regression (GPR) (65), which is a well-studied model in Bayesian Optimization (BO) field (76) and is adapted by several analog sizing algorithms. The main drawback of GPR modeling is the computational complexity involved in building models.

Recent research trend in analog sizing introduces Machine Learning (9) to simulation-based methodology. However, the literature review reveals that most of these methods require thousands of simulation data to train Deep Neural Network (DNN) models that approximate the relations between the design variables and the performance metrics. Therefore, the practicality of these algorithms is severely reduced when the optimization task has a high simulation cost. For example, drawing/generating the layout, extracting the parasitics, and running post-layout simulations is typically an expensive procedure. Therefore, optimization algorithms designed for schematic-level sizing can not be adapted by simply changing how data is generated.

1.1.1 Schematic-Level Sizing Automation

Analog integrated circuit (IC) sizing has been investigated for decades. Among the available methods, a widely accepted one is simulation-based global optimization (45; 22). In the optimization process, the candidate designs are evaluated via SPICE simulations. In the past, advanced optimizers for ana-

log IC sizing (2; 18; 4; 91; 44) have been proposed to handle design cases with stringent specifications and obtained successful results. Hence, in recent years, the research focus gradually transformed from "effective" sizing to "efficient" sizing (56; 39). The recent efficient methods for AMS sizing can be collected under two algorithm classes: Bayesian Optimization methods and Deep Learning methods.

Bayesian Optimization (BO) methods are tested on AMS problems and are proven to be sample efficient. For example, GASPAD (41) is a hybrid algorithm using a combination of evolutionary space exploration and GPR surrogate-based selection. WEIBO (57) method also employs GPR as a surrogate and introduces a Bayesian Optimization framework where a weighted acquisition function is tailored to comply with the performance-constrained nature of the sizing problem. In (29), the authors introduced a multi-fidelity GPR algorithm where the fidelity of the performance is varied with the simulation accuracy. However, this work did not address the layout effects. The disadvantage shared by all GPR models is their cubic complexity to the number of samples, $\mathcal{O}(N^3)$.

Deep Learning based sizing methods include supervised learning and reinforcement learning (RL) methods (5; 86; 69; 6; 49). GCN-RL (86) is a Graph Neural Network algorithm where state representation is built via device index, type, and selected electrical properties. They also propose methods

to transfer the optimization experience between different topologies and processes. However, their training graphs show that they use up to 10^4 simulations for sizing academic circuits. AutoCkt (69) is a discrete action space policy gradient method. The RL agent is trained on different optimization tasks where the task is randomly sampled from a predefined set. The trained agent is then tested for the particular tests during deployment. We also observe from the training graphs that AutoCkt requires up to 10^5 simulated samples for training. In (23), the authors successfully applied BNN on multi-objective analog sizing. However, they did not consider handling constraints, and layout effects are ignored. DNN-Opt (6) is introduced as an RL-inspired supervised learning optimization method that shows high sample efficiency and can be trained during optimization. It uses less than a thousand iterations to optimize academic benchmarks. DNN-Opt does not quantify the variance on approximated values and has no methodic way to balance exploration/exploitation during design space exploration.

1.1.2 Layout-Aware Sizing Automation

Several works in AMS sizing proposed solutions to include layout-induced parasitics. The studies proposed in (54) and (28) embedded a layout generator in the automation loop, and performance metrics to be optimized are obtained through post-layout simulations. However, they did not

consider the correlations between the schematic-level and post-layout simulations; therefore, their efficiency is limited. The work in (51) employs a less accurate parasitic prediction during sizing, so the finalized post-layout performance is not guaranteed. In (49), the authors propose a Transfer Learning strategy where a DNN is first trained on schematic-level simulations. Then this knowledge is transferred to improve the learning of a relatively small number of post-layout data. Although this work suggests improving the efficiency of post-layout optimization, it requires up to 5×10^3 schematic-level data for initial DNN training, which suffers from the scalability concerns mentioned before. In summary, a scalable solution to optimize AMS design parameters under layout parasitics is yet to be studied.

1.2 Overview of Dissertation

This dissertation includes several algorithms and methodologies for the AMS IC sizing problem. We primarily focus on applying Machine Learning (ML) to optimize the performance of AMS circuits efficiently.

Chapter 2 introduces an ML-assisted global optimization framework, (5). A pseudo-sample generation method is introduced to increase the number of samples for model training. The generated data is used to train a feed-forward neural network to approximate the circuit performance. Then, we introduce a hybrid search strategy where candidate solutions are generated via evolution-

ary algorithm steps. We improve search efficiency by using the neural network as a proxy to the real simulator.

Chapter 3 presents a Reinforcement Learning (RL) inspired optimization method for AMS sizing, (6). We borrow deep actor-critic algorithm studied by the RL community and tailor it for the AMS sizing task. We identify how to define the state, actions, and return for the AMS sizing problem. We further introduce a recipe for applying an RL-based optimizer on very large circuits with thousands of devices. We demonstrate the performance of our approach both on academic circuits and industrial circuits. We observed a significant performance improvement compared to several conventional baseline algorithms and compared to existing commercial tools.

Chapter 4 focuses on AMS tasks with varying simulation costs(8). We leverage cheap-to-run simulations to make intermediate decisions on the potential quality of explored points. Then we refrain from expensive-to-run simulations if necessary. In addition, we introduce an asynchronously parallel framework and adapt our previous work for the case of designs with the differentiated cost of simulations. Our benchmarking shows that the proposed methods significantly reduce the total real-time cost of optimization and it also reduces the total CPU effort.

In Chapter 5, this dissertation includes a solution for effectively solving the sizing problem under layout effects(10; 11). We conducted a case-study to

quantify the effects of considering layout during transistor sizing. Then, we applied a Bayesian Neural Network (BNN) based approach to solve the sizing problem. To include layout-induced parasitics, we extend our approach via Multi-Fidelity BNN, where the algorithm utilizes multiple information sources to learn post-layout performances efficiently. We also include a search-space exploration strategy using the trust-region approach which is shown to be effective on problems with large number of input dimensions. Our tests suggest that the BNN-based sizing algorithm is very competitive compared to previous state-of-the-art algorithms. We further demonstrate that the co-learning strategy of Multi-Fidelity BNN further improves the efficiency, which is very crucial considering the costly post-layout simulations.

Finally, Chapter 6 summarizes and concludes this dissertation and also discusses the potential future research topics.

Chapter 2: An Efficient Analog Circuit Sizing Method Based on Machine Learning Assisted Global Optimization[1]

2.1 Introduction

One of the main reasons to motivate the efficiency improvement of analog IC sizing is the increasingly stringent design specifications. For a common analog building block with moderate specifications, manual design is also effective and efficient. In contrast, building blocks with stringent design specifications often need many simulations to obtain a satisfactory design in optimization (45; 56), and the accumulated simulation time could be long. In particular, some analog circuit simulations involve Monte-Carlo analysis (e.g., periodic noise, transient noise) and are computationally expensive.

An effective way to improve optimization efficiency is by introducing machine learning techniques into optimization (56; 47; 41; 37; 24; 53; 25). Surrogate models, which are often constructed by machine learning techniques,

[1]This chapter is based on the following publication: Ahmet Faruk Budak, Miguel Gandara, Wei Shi, David Z Pan, Nan Sun and Bo Liu, "An Efficient Analog Circuit Sizing Method Based on Machine Learning Assisted Global Optimization" in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD) 2021. I am the main contributor in charge of problem formulation, algorithm development and experimental validations.

are ideally computationally cheap approximation models of simulations. In optimization, they are employed to replace simulations to save time. Another effective way is employing parallel computing. (39) realizes the co-use of surrogate model-based optimization and parallel computing in its local search phase, and the global exploration phase is implemented by a parallel evolutionary algorithm. Because parallel analog IC sizing itself is a research area (e.g., parallel simulation of different candidate designs, parallel various kinds of simulations, parallel samples in periodic noise simulation), which depends on the computing capacity and the particular problem, it will be studied separately and is out of the scope of this chapter.

A surrogate model-assisted optimization-based analog IC sizing method often has three main elements: the optimization algorithm, the surrogate modeling method, and the infill sampling criterion. For optimization algorithms, either evolutionary algorithms (41) and multi-start local optimization algorithms (91; 56) are employed. Regarding surrogate modeling methods, many existing research works use the Gaussian process (GP) (66) because of its strong learning ability compared to other alternatives (e.g., standard artificial neural networks (ANN), radial basis functions) (56; 41; 43; 30; 93). Therefore, typical methods for microwave analog circuit synthesis, such as GASPAD (41) and typical methods for general analog circuit sizing, such as WEIBO (56), employ GP.

However, the computational cost of GP modeling is not negligible. The computational complexity is $O(N^3)$ (20), where N is the training data points to construct a reliable surrogate model. In addition, the number of input variables is a non-dominating but important factor for the GP modeling complexity. The works in (30; 93) propose new methods to reduce the GP training cost, which are important. However, when considering the complete set of specifications, which could be more than 20, the GP modeling may still be a burden compared to working with a few specifications. This considerably cancels out the time saved by the reduced number of simulations.

Infill sampling criterion (60) is also essential for the success of a surrogate model-assisted analog IC sizing method. The infill sampling criterion investigates how to make use of the predicted value and prediction error to obtain a high-quality ranking of candidate designs in order to guide the optimization engine. For example, the efficient global optimization technique (33), where the expected improvement infill sampling method plays a key role, is introduced into analog IC sizing (19). Also, the weighted expected improvement (wEI) (26) and lower confidence bound (20) methods are used in WEIBO (including its improvements) and GASPAD, respectively. However, when considering the complete set of specifications, we found that both infill sampling criteria suffer. This is because (1) some analog IC performances are not easy to learn due to their nature and simulation failure, and (2) the accumulated

prediction error of the complete set of performances adds more uncertainty, which can easily mislead the optimization engine, even using GP. This may fail the sizing, which is particularly clear when the specifications are stringent (Section IV).

To address the above challenges when considering the complete set of specifications, a new method, called Efficient Surrogate Model-assisted Sizing Method for High-performance Analog Building Blocks (ESSAB), is proposed in this chapter. The key innovations include: (1) an infill sampling criterion only using the predicted value, which is therefore robust to the possibly large prediction error, (2) a new ANN model construction method avoiding using GP but obtaining even better prediction quality, using which, the machine learning cost no longer becomes a problem. A surrogate model-assisted evolutionary algorithm (SAEA) framework is then proposed to make use of them. Experiments using the complete set of stringent specifications verify the advantages of ESSAB.

2.2 Problem Formulation

In this dissertation, we assume that the existence of post-layout performance implies the existence of schematic-level performance values. However, the reverse implication does not hold. We formulate the AMS schematic-level sizing and layout-aware sizing task as a constrained optimization problem suc-

cinctly as below.

$$\begin{aligned} &\text{minimize } f_0(\mathbf{x}) \\ &\text{subject to } f_i(\mathbf{x}) \leq 0 \quad \text{for } i = 1, \ldots, m \end{aligned} \tag{2.1}$$

where, $\mathbf{x} \in \mathbb{R}^d$ is the parameter vector and d is the number of design variables of sizing task. Thus, $\mathbb{R}^d$ is the design space. $f_0(\mathbf{x})$ is the objective performance metric we aim to minimize. Without loss of generality, we denote i^{th} constraint by $f_i(\mathbf{x})$. Notice that if the problem is schematic-level optimization, the f_i values are obtained from schematic simulations. If the problem is post-layout optimization, the f_i values are determined by post-layout simulations.

Through this dissertation, we will evaluate the quality of a design by defining a Figure of Merit (FoM) in the following form:

$$\text{FoM}(\mathbf{x}) = w_0 \times f_0(\mathbf{x}) + \sum_{i=1}^{m} \min\left(1, \max(0, w_i \times f_i(\mathbf{x}))\right) \tag{2.2}$$

where w_i is the weighting factor. Note, a $\max(\cdot)$ clipping is used for equating designs after constraints are met, and $\min(\cdot)$ is used to prevent single constraint violation from dominating FoM value.

2.3 Background

In this section, we provide a review of related concepts used to build ESSAB.

2.3.1 Differential Evolution

Differential evolution (DE) (79; 62) is a popular global optimization algorithm. The mutation and crossover operators in the DE algorithm is adopted in ESSAB, which works as follows.

Let P be a population composed of a number of individual solutions $x = (x_1, \ldots, x_d) \in R^d$. To generate a child solution $u = (u_1, \ldots, u_d)$ for x, a donor vector is first produced by mutation (the DE/current-to-best/1 strategy is used in this work):

$$v^i = x^i + F \cdot (x^{best} - x^i) + F \cdot (x^{r_1} - x^{r_2}) \tag{2.3}$$

where x^i is the i^{th} vector in the current population and x^{best} is the best candidate in the current population P, x^{r_1} and x^{r_2} are mutually exclusive solutions randomly selected from P (the current population); v^i is the i^{th} mutant vector in the population after mutation; $F \in (0, 2]$ is a control parameter, often called the scaling factor.

Then the following crossover operator is applied to produce the child u:

1 Randomly select a variable index $j_{rand} \in \{1, \ldots, d\}$,

2 For each $j = 1$ to d, generate a uniformly distributed random number $rand$ from $(0, 1)$ and set:

$$u_j = \begin{cases} v_j, & \text{if } (rand \leq CR) | j = j_{rand} \\ x_j, & \text{otherwise} \end{cases} \tag{2.4}$$

where $CR \in [0, 1]$ is a constant called the crossover rate.

2.3.2 Artificial Neural Networks

ANN (88) is a widely used learning machine for surrogate modeling and prediction. The structure of ANN mimics the process of knowledge acquisition, information processing, and organizational skills of the human brain. Hence, it is able to learn complex nonlinear relationships from a training data set. Among various kinds of ANNs, the feedforward ANN is used in this research.

The structure of a typical feedforward ANN is shown in Fig. 2.1, which is composed of a number of highly interconnected neurons. With n-dimensional input data points (i.e., input layer) and l-dimensional output data points (i.e., output layer), a hidden layer with m neurons is used.

Signals generated from the input layer propagate through the network on a layer-by-layer basis in the forward direction. Each neuron accepts output data from neurons in the previous layer using different weights and adjusts the weighted sum by its activation function, to generate the output. To achieve the final desired outputs (i.e., minimize the error between the predicted outputs and the desired outputs), the thresholds and weights, which are controlled by the level of the activation of each neuron, and the strength of the connections

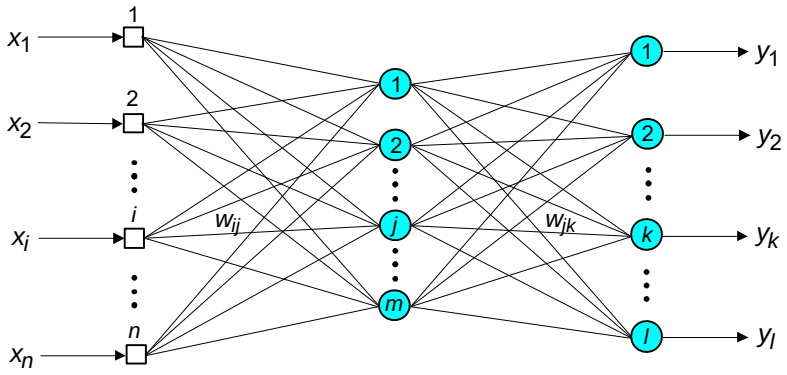


Figure 2.1: A feedforward ANN with one hidden layer

between the individual neurons, are trained. When the average error is within a predefined tolerance, the training terminates, and the weights are locked in; the network is then ready to be used for predicting new inputs. The performance of ANN is largely related to its structure. In terms of training and prediction cost, the ANN is computationally much cheaper compared to GP.

2.4 Methodology

Analog IC sizing with the complete set of specifications has the following characteristics: (1) The number of specifications could be 10 to 20 or even more. Particularly, for amplifiers, the saturation margin specifications could be many and are necessary to be included in the optimization. (2) Some specifications are not easy to learn. For example, some noise performances

are highly nonlinear, the settling time may not be found for many candidate designs because they never settle in the transient analysis time window. Because the above characteristics are different from benchmark problems used in the computational intelligence field, to the best of our knowledge, there is no off-the-shelf surrogate model-assisted optimization algorithm considering this.

As indicated in the previous section, two challenges are brought by the above characteristics, which are: (1) Inaccurate prediction and the accumulation of the prediction error mislead the optimization engine and fail the sizing, especially when the specifications are stringent. (2) The machine learning time can be another burden that cancels out the saved simulation time to a large extent. Note that ESSAB does not aim at providing a complete solution for analog IC sizing. Instead, it aims at addressing the above two key challenges when introducing machine learning techniques into optimization-based analog IC sizing. Important research topics such as sizing problem definition assessing the merit of designs, yield optimization considering process, voltage and temperature variations (83) are out of the scope but compatible with ESSAB.

2.4.1 Overall Flow

The flow diagram of ESSAB is shown in Fig. 2.2. The algorithm works as follows.

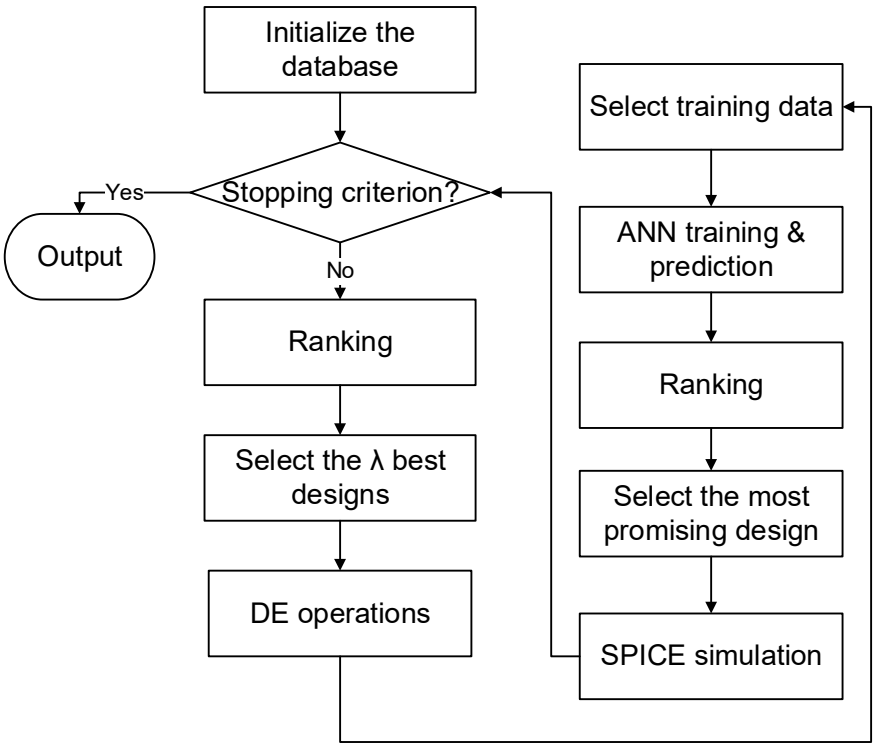


Figure 2.2: The flow diagram of the ESSAB method

Step 1: Sample α (often a small number) candidate designs from the design space $[a, b]^d$ (a and b are the lower and upper bounds of the design variables, respectively; d is the number of design variables) using the Latin Hypercube sampling method (78). Carry out SPICE simulations and form the initial database.

Step 2: If a preset stopping criterion is met, output the best design from the database; otherwise, go to Step 3.

Step 3: Rank all the designs in the database using the infill sampling criterion in Section III (B). (Simulation results are used in the ranking process.) Select λ best candidate designs to form a population P.

Step 4: Apply the DE mutation (Eq. 2.3) and crossover operator (Eq. 2.4) on P to generate λ child solutions.

Step 5: Select τ best solutions in the database and their performances as the training data. Construct an ANN model using the method in Section 2.4.3.

Step 6: Rank the λ child solutions using the infill sampling criterion in Section 2.4.2. (ANN-predicted results are used in the ranking process.)

Step 7: Simulate the estimated best child solution from Step 6. Add this solution and its performance values from the simulation to the database. Go back to Step 2.

ESSAB flow suggests that it is an online surrogate model-assisted global optimization method (45), and there is no preliminary training data points or surrogate model. In Step 1, the α initial sampling is often small (e.g., $5 \times d$ by

default), and a very coarse surrogate model is constructed. In each iteration, the surrogate model quality and optimal designs are gradually improved. This is in contrast with offline surrogate model-assisted optimization, which first constructs a relatively accurate surrogate model for the substitution of SPICE simulations; a few or no simulations are carried out in the optimization process. The main challenge for offline surrogate model-assisted optimization is that considerable efficiency improvement is difficult to be maintained when the number of design variables is larger than a few (47; 48). A linear increase in the number of design variables leads to an exponential increase in the design space. To maintain a reasonably good surrogate model accuracy, much more samples are needed with a small increase in the number of design variables, which soon becomes tremendous. However, only a small region of the design space is useful in optimization, and many of these simulations to cover the whole design space (most are not near the optimal region) are wasted.

Among various online surrogate model-assisted optimization frameworks, ESSAB follows the surrogate model-aware evolutionary search (SMAS) framework (47; 42). In this framework, the selection operators are unique. The λ current best candidate designs are selected to form the parent population (it is reasonable to assume that the search focuses on the promising region), and the best-predicted candidate design in the child population is selected to replace the worst one in the parent population. Hence, only at most one

candidate solution is changed in the parent population in each iteration. The best candidate designs in the child solutions in several consecutive iterations are likely near to each other, which will be simulated and used as training data points. Hence, the training data points describing the current promising region can be much denser compared to those generated by a standard EA population updating mechanism, which may spread in different regions of the design space. This largely improves the surrogate model quality (42; 1).

However, borrowing the SMAS framework is insufficient to address the above two targeted challenges for analog IC sizing. In ESSAB, the two innovations are ranking and surrogate modeling. The ranking operator (or infill sampling criterion) is one of the keys for any SAEA, which guides the optimizer to find the feasible region in the design space and then the optimal design. In ESSAB, it is used in Step 3 to determine the parent population for the current iteration and in Step 6 to determine the single candidate design for simulation considering the predicted values of the child population. ANN modeling is another key operator because the prediction accuracy highly determines the quality of ranking in Step 6, which is also essential to guide the optimization engine. Moreover, machine learning cost is mainly determined by the ANN training. The details of the above two key operators are described in the following subsections.

2.4.2 Parameter-Free Ranking

Before introducing the proposed ranking method, it is worth analyzing the drawbacks of existing infill sampling criteria when considering the complete set of specifications. Traditional infill sampling criteria, such as expected improvement (33), lower confidence bound (20), do not consider constraints. Instead, they aim to estimate the quality of a solution based on both the predicted value and the prediction uncertainty, in contrast to only considering the predicted value. When handling constraints, they often need to be combined with the penalty function method (63). A typical example is the GASPAD algorithm for mm-wave IC synthesis (41). Such methods often work well when there are a few specifications, as mm-wave IC synthesis problems have. However, it is known that for analog IC sizing problems, the penalty coefficients are often sensitive, especially when the design specifications are stringent (44). Moreover, the number of specifications can be many (e.g., more than 20), instead of a few.

There are several recent works that consider constraints in their infill sampling criteria. For example, in the weighted expected improvement (wEI) criterion, the expected improvement value for the objective function is multiplied by the probability of feasibility of all constraints (26). This method does not need penalty coefficients. However, the basic assumption is that the prediction error is reasonable, which can provide useful information for de-

termining the overall quality of a candidate design. As said above, for some analog IC performances, the predicted value may still be reasonable using GP, but the prediction uncertainty (i.e., error) can be large. Also, considering the prediction error is accumulated by many performances, wEI becomes less effective since the estimated probability of feasibility may be affected.

Therefore, two principles of the new infill sampling criterion are: (1) It considers all the constraints, and no penalty coefficient is used. (2) It avoids using prediction errors and will therefore not be affected by some large prediction errors and the accumulation of them. With these basic ideas, a new method is developed. We call it the probability of further improvement (PFI) criterion, which works as follows.

Considering that there are n designs to be ranked, for each of them, there are m performances $y_j, j = 1, 2, \ldots, m$ (either simulated values or predicted values). Among the m performances, one of them serves as the objective function, and others are only constraints. They can be described as $c(x) \leq S_j, j = 1, 2, \ldots, m$, where S_j is the j_{th} specification. An $n \times m$ performance matrix can be formed. The PFI infill sampling criterion ranks the n designs as follows (Algorithm 1).

Some clarifications are as follows:

- The proposed PFI infill sampling criterion is directed by the probability

Algorithm 1 The PFI infill sampling criterion

for all performance j **do**:
    Normalize the j_{th} column of the performance matrix and fit it into a β distribution, Beta(α_j, β_j), to obtain the hyperparameters α_j and β_j;
    For S_j, obtain cumulative distribution function CDF(S_j) with fitted hyperparameters;
    For each design i $(i = 1, 2, \ldots, n)$, calculate the probability of obtaining a better performance than the current y_j^i while still violating S_j:
    if $y_j^i > S_j$ **then**
        $B_j^i = \text{CDF}\left(y_j^i | \,\text{Beta}(\alpha_j, \beta_j)\right) - \text{CDF}\left(S_j | \,\text{Beta}(\alpha_j, \beta_j)\right)$
    else
        $B_j^i = 0$
    end if
end for
For each candidate design, calculate the potential value by
For each candidate design i $(i = 1, 2, \ldots, n)$, calculate the potential value by

$$Po\,(i) = \sum_{j=1}^{m} B_j^i \tag{2.5}$$

Rank the n candidate designs based on their Po value in ascending order (i.e., the smaller the better) and break the ties based on the objective metric.

of further improvement based on the current visited designs and performances. Besides considering performance improvement, satisfying the constraint is considered in this process, which is reflected in the calculation of B_j^i.

- It can be seen that PFI avoids using the prediction error and it also

does not need penalty coefficients, which is in line with the principles mentioned before.

- The reason why Beta distribution is used is that the PFI requires a probabilistic interpretation of the members in the performance matrix. Beta distribution provides the flexibility to capture the irregular distributions of the fitted performance values.

2.4.3 Pseudo-Sample Generation and Training

Since PFI does not use prediction error to evaluate the potential of a candidate design, it requires the prediction value to be sufficiently accurate. Our initial experiments found that the GP model can satisfy the accuracy requirement, but the GP modeling time is long in analog building block sizing. On the other hand, ANN training is efficient, inspiring us to improve ANN's prediction ability.

A feedforward ANN model with a single hidden layer is widely used for regression tasks. Experiments show that the accuracy of using that model is insufficient for the targeted analog IC performances. An effective way to improve prediction accuracy is to use more training data points and hidden layers. However, in machine learning-assisted analog IC sizing, the goal is to reduce the number of simulations. The number of training data points is, therefore, only a few. In addition, candidate designs near the current search

region (i.e., the current best τ designs according to Section 2.4.2) provide essential information to predict the subsequent population, while most other candidate designs do not contribute much. This increasingly decreases the number of available training data points.

In ESSAB, the key idea is to increase the number of training data points. Given the original training data set (X), which are the τ current best designs obtained so far, two samples $(x^i, y^i) \in R^d$ and $(x^j, y^j) \in R^d$ are randomly selected. The corresponding point in the new training data set is

$$x^{ij} = (x^i,\ \Delta x^{ij}) \text{ and } y^{ij} = y^j \tag{2.6}$$

where $\Delta x^{ij} = x^j - x^i$. Following this way, a new training data point can be generated for each ordered pair in the original training data set. Formed by the cartesian product of X of size τ with itself, which is given as

$$\mathrm{X} \times \mathrm{X} = \{(x^i, x^j) \mid x^i \in \mathrm{X} \text{ and } x^j \in \mathrm{X}\}$$

the number of new training data points is τ^2.

The ANN model is constructed using the new training data set with $2d$ dimensions as shown in Figure 2.3.

Using this ANN model with two hidden layers, the weights are trained using the loss function, calculating the distance between the ANN predicted

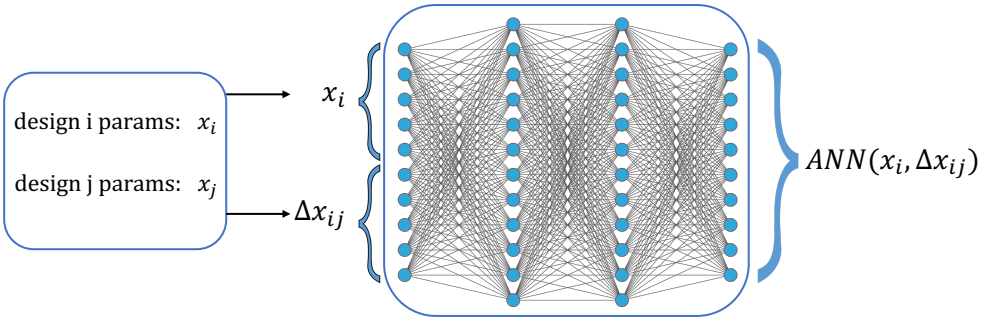


Figure 2.3: The ANN model in ESSAB

and simulation values.

$$L(w) = E\left((ANN(x, \Delta x) - y\left(x + \Delta x\right)\right)^2\right) \tag{2.7}$$

where $ANN(x, \Delta x)$ are the ANN predicted values, and $y(x + \Delta x)$ are the performances obtained by simulations. The ANN is trained using the ADAM optimizer (35). The learning rate is 0.001, and the batch size is 64. For hidden layers, the rectifier activation function is used, and there is no activation function for the output layer. Often, 10 training epochs are sufficient.

After the ANN model is trained, in Step 6 of the ESSAB framework (Section 2.4.2)), the constructed ANN model with $2d$ inputs is used to predict the child population generated by DE operators with d inputs. The λ child solutions generated in Step 4 is used as x^j and the λ solution obtained in Step 3 is used as x^i. Following Equation 2.6, the input values for the ANN model can be generated. Their performance can then be predicted.

2.4.4 Parameter Settings

ESSAB has SAEA parameters and ANN parameters. Regarding the former, the settings are as follows: $\alpha = 5 \times d$, $\lambda = 5 \times d$, $\tau = 5 \times d$, where d is the number of design variables. These settings follow the general SAEA parameter setting principles (e.g., (45)). F and CR are DE parameters. They are set as $F = 0.8$, $CR = 0.5$, following (79). In terms of ANN parameters, 2 hidden layers are used, and each of them has 100 neurons. These are empirical settings by studying data characteristics of analog IC design performances, and once set, they do not change.

2.5 Experiments

In this section, the performance of ESSAB is verified by four test cases, including a two-stage folded-cascode operational transconductance amplifier, an inverter stacking amplifier (72), a strong-arm latch comparator (81) and a distributed-input voltage controlled oscillator (VCO) (58). 180 nm CMOS technologies are used for the first three test cases and a 40 nm CMOS technology is used for the last test case. Note that the complete set of stringent specifications used in real-world design practice are used for test cases 1, 3 and 4. The specifications come from DC, AC, and transient analyses, having relations with each other. According to applications, some performance is more important than others. For generality consideration, all the specifications are

considered as important. For test case 2, the specifications in (72) are followed for comparison purposes, where transient analysis-based specifications are not used. The number of specifications used in the case studies is up to 29. Besides the large number, some performances are not easy to learn as said in Section 2.4, which appear in test cases 1, 3 and 4. All the experiments are run on a workstation with Intel Xeon CPU and 128GB RAM. Cadence Spectre is the simulation tool. No parallel computing is considered.

Because the two major innovations of ESSAB are the new infill sampling criterion PFI and the new ANN model construction method, the following two reference methods are proposed. The first one is ESSAB-GP, which replaces the ANN prediction with GP prediction. The aim is to observe the prediction ability and training cost of the two machine learning methods. The second one is Bayesian optimization using wEI (BO-wEI). Bayesian optimization is a surrogate model-assisted optimization method, for which infill sampling plays a key role and many of them are based on GP modeling. The framework follows (56), where GP and wEI are used. The difference compared to (56) is that multi-start local optimization is replaced by DE considering the time consumption of GP predictions when using sufficient starting points. This reference method aims to observe the ranking ability of wEI when the complete set of specifications (some of them are difficult to learn) are used. As a benchmark reference method for analog IC sizing, DE is compared with ESSAB in

terms of both solution quality and efficiency. Besides, in the second, third and fourth test cases, the results are also compared with published designs (72; 81; 58). Both the results from (72; 81; 58) and ESSAB are based on simulation.

Integer values are involved in all three test cases. The same quantization method is used for all the algorithms as in (62; 43). Random numbers are involved in stochastic algorithms. Hence, 10 runs are carried out for each algorithm, and the results are analyzed statistically. To converge the algorithms, the simulation budget for test cases 1, 2, and 3 is 10,000 for DE and 500 for ESSAB and BO-wEI. For test case 4, because each simulation costs more than 10 minutes, DE is expected to cost prohibitive amount of time and is not used. The simulation budget for ESSAB and BO-wEI is 200 simulations making them converge. For ESSAB-GP, because the goal is to verify the efficiency and quality of the novel ANN model, the same number of simulations are used for a fair comparison with ESSAB, although the algorithm does not fully converge. Using more simulations, making the algorithm converge is carried out separately.

Some of the reference methods cannot satisfy the specifications. Hence, there is no feasible solution, and comparing objective function values for feasible solutions is impossible. To show the performance of different algorithms, a metric is defined for a candidate design before satisfying all the specifications.

We call it performance improvement indicator (PII) as follows.

$$\text{PII} = \sum_{j=1}^{m} \min\left(1, \max\left(0, \frac{y_j - S_j}{y_j^{\text{ref}} - S_j}\right)\right) \tag{2.8}$$

where $y_j (j = 1, \ldots, m)$ is the j_{th} performance of the candidate design, S_j is the j_{th} specification (Section III) and y_j^{ref} is a reference point, which is defined by the designer for normalization. y_j^{ref} refers to least workable value in general cases. For example, when S_j for DC gain is set to 70 dB, y_j^{ref} can be set to 40 dB. (Note that all the specifications use $c(x) \leq S_j$ in PII. S_j is -70 dB and y_j^{ref} is -40 dB considering -DC gain $\leq$ -70 dB). The value of y_j^{ref} does not need to be accurate and is case-dependent. Although using different y^{ref} may lead to a slight change in the PII-based convergence curve, the same y^{ref} is used for all the reference methods for a fair comparison. In PII, max() is used considering constraint satisfaction instead of objective function optimization; min() is used to prevent a single specification dominates the PII value. Hence, the worst value of PII for a candidate design is m, indicating that the candidate design is worse than or equal to the design meeting the least common workable values but far below satisfactory, while the best value of PII is 0, indicating all the specifications are satisfied. Notice that PII formulation in this chapter shares the same principle with FoM defined in Equation 2.2.

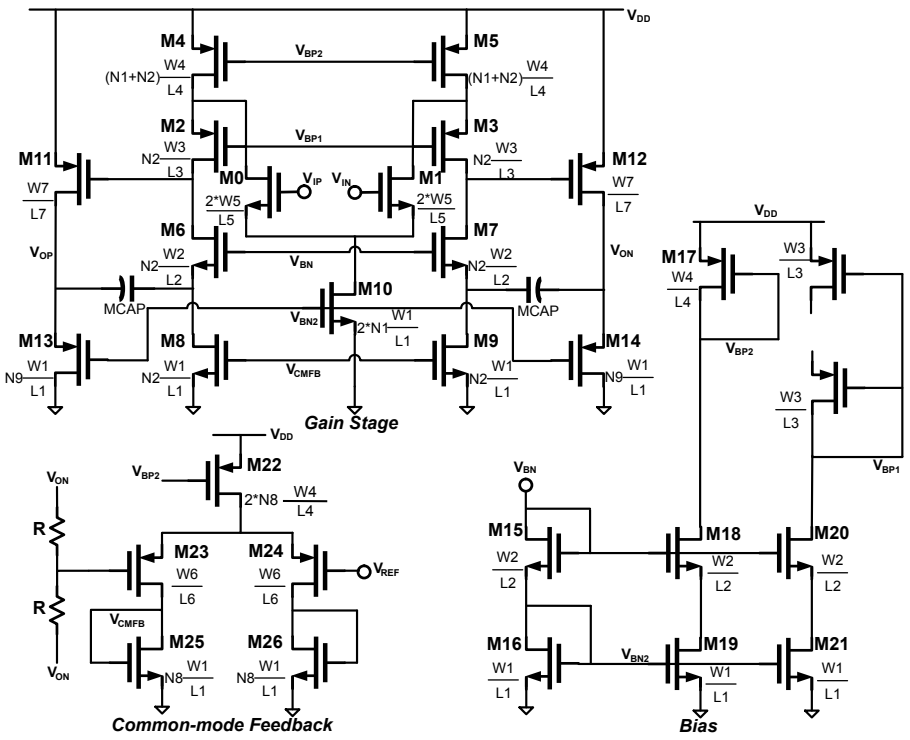


Figure 2.4: Schematic of the folded-cascode OTA

2.5.1 Case Study 1: Folded-Cascode OTA

The first test case is a folded-cascode operational transconductance amplifier (OTA) (Fig. 2.4). It has 20 design variables, and the search ranges provided by the designer are in Table 2.1.

Table 2.1: Design parameters and their ranges for the folded-cascode OTA

Parameter	LB	UB	Parameter	LB	UB
L1(μm)	0.18	2	W4(μm)	0.24	150
L2(μm)	0.18	2	W5(μm)	0.24	150
L3(μm)	0.18	2	W6(μm)	0.24	150
L4(μm)	0.18	2	W7(μm)	0.24	150
L5(μm)	0.18	2	MCAP(fF)	100	2000
L6(μm)	0.18	2	Cf(fF)	100	10000
L7(μm)	0.18	2	N1 (integer)	1	20
W1(μm)	0.24	150	N2 (integer)	1	20
W2(μm)	0.24	150	N8 (integer)	1	20
W3(μm)	0.24	150	N9 (integer)	1	20

W: transistor width; L: transistor length; UB: upper bound; LB: lower bound

The sizing problem is defined as follows:

$$
\begin{aligned}
&\text{minimize } \mathit{Power} \\
&\text{s.t.} \quad \text{DC Gain} \geq 60 \ \text{dB} \\
&\qquad \text{CMRR} \geq 80 \ \text{dB} \\
&\qquad \text{PSRR} \geq 80 \ \text{dB} \\
&\qquad \text{Output Swing} \geq 2.4 \ \text{V} \\
&\qquad \text{Output Noise} \leq 3 \times 10^{-4} \ \text{V}_{\text{rms}} \\
&\qquad \text{Phase Margin} \geq 60 \ \text{deg} \\
&\qquad \text{Unity Gain Frequency} \geq 30 \ \text{MHz} \\
&\qquad \text{Settling Time} \leq 3 \times 10^{-8} \ \text{s} \\
&\qquad \text{Static Error} \leq 0.1\% \\
&\qquad \text{Saturation Margins} \geq 50 \ \text{mV}
\end{aligned}
\tag{2.9}
$$

In our experiment, the following transistors are required to operate in the saturation region: M1, M3, M4, M7, M9, M10, M12, M13, M15, M16, M17, M18, M19, M20, M21, M22, M23, M24, M25 and M26. The total number of specifications becomes 29. Note that the saturation margin specifications are essential. Solutions satisfying all the performance specifications can be found much easier without considering them, but some transistors among M3, M4, M10, and M22 (Fig. 2.4) are in the triode region. A similar phenomenon also applies to the next test case, which will not be repeated. Note that different objective functions and constraints other than (2.9) can be used. For example, (83) defines a new kind of Figure of Merit. ESSAB applies to any objective functions and constraints.

Ten runs are carried out for ESSAB and all of them satisfy all the specifications. ESSAB converges within 400 simulations, which is fewer than the simulation budget (500 simulations). The total sizing time is about 2.5 hours (system time). A typical design obtained by ESSAB is shown in Table 2.2 (including current biases) with the corresponding performance in Table 2.3.

The statistical results for all the reference algorithms are shown in Table 2.4. In Table 2.4, the success rate is the number of runs that obtain feasible designs (i.e., satisfying all the specifications) in the given simulation budget divided by the total 10 runs. $N_{feasible}$ is the number of simulations used to obtain the first feasible design (average over 10 runs). The statistics

Table 2.2: A typical design obtained by ESSAB (test case 1)

Parameter	Value	Parameter	Value	Parameter	Value
L1(μm)	1.28	L2(μm)	0.36	L3(μm)	0.18
L4(μm)	1.9	L5(μm)	0.4	L6(μm)	1.7
L7(μm)	0.48	W1(μm)	6	W2(μm)	9.6
W3(μm)	6	W4(μm)	126	W5(μm)	142
W6(μm)	40.4	W7(μm)	132	MCAP(pF)	1.8
Cf(pF)	1.3	N1	4	N2	5
N8	1	N9	6	Id22 (nA)	240
Id10 (nA)	80	Id13/Id14 (nA)	200	Id3/Id4 (nA)	60

Table 2.3: Performance values of a typical design obtained by ESSAB (test case 1)

Power 0.63 mW	DC Gain 96.5 dB	CMRR 96.5 dB
PSRR 138.1 dB	Output Swing 2.69 V	Output Noise $2.72 \times 10^{-4} V_{rms}$
Phase Margin 77 deg	Unity Gain Freq. 34 MHz	Settling Time 2.5×10^{-8} s
Static Error 0.004%	Saturation Margins all satisfied	

of the objective function value (i.e., power) only consider the feasible runs. The modeling time is the total system time of GP or ANN modeling and

Table 2.4: Statistical results for different algorithms (test case 1)

Algorithm	DE	BO-wEI	ESSAB-GP	**ESSAB**
Success rate	10/10	2/10	10/10	**10/10**
N_{feasible}	3600	N.A.	252	**160**
Min. power (mW)	0.82	0.91	0.79	**0.53**
Max. power (mW)	1.55	1.62	1.12	**0.86**
Mean power (mW)	1.18	1.25	0.96	**0.68**
Std. power (mW)	0.33	0.5	0.12	0.09
Modeling time (h)	N.A.	29	6.5	0.4
Simulation time (h)	52	2.6	2.6	2.6

prediction in a single run (average over 10 runs). The simulation time for BO-wEI, ESSAB-GP, and ESSAB are very similar, although slightly different. Because the same kind of simulation is carried out for the same number of times, the average simulation time, 2.6 hours, is used for all of them. This also applies to other test cases.

The following conclusions can be drawn from Table 2.4. The designs obtained by ESSAB are of high performance, even for the worst case. The average power value is the best among all reference methods. The modeling time (0.4 hours) is very short, successfully addressing the key challenge of machine learning cost.

Compared to DE, a benchmark reference method for analog IC sizing,

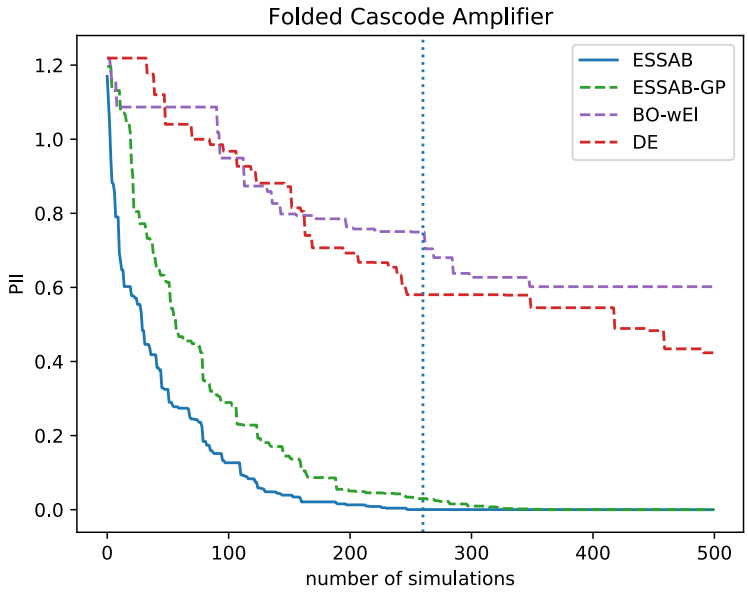


Figure 2.5: Folded Cascode PII Values (average over 10 runs)

ESSAB obtains better solution quality. In terms of efficiency, ESSAB can obtain the first feasible design using almost 22 times fewer simulations than DE. Even considering that DE does not have modeling time, ESSAB is much more efficient than DE due to the largely reduced number of simulations and the short modeling time.

ESSAB-GP shows the performance of the ANN model compared with the GP model in the same algorithm framework. Within 500 simulations, although both ESSAB and ESSAB-GP obtain 100% success to satisfy all the specifications, ESSAB outperforms ESSAB-GP. To obtain the first feasible design, ESSAB-GP needs 252 simulations on average, while ESSAB only needs

160 simulations. ESSAB also shows advantages in the objective function values compared to ESSAB-GP. This shows that the new ANN model construction method is better than GP in terms of prediction quality and efficiency. In GP-based methods, the GP modeling and prediction time is often much more than the simulation time (even though they are more efficient than DE), while the new ANN model training and prediction cost about 15% of the simulation time.

BO-wEI runs show that most of the specifications are satisfied, but in 8 cases, the output swing, settling time, and a few saturation margin specifications are often violated. Note that BO-wEI is often trapped in local optima, and the performance will not be improved when using more simulations. This can also be found in the PII value (Fig. 3.2). The worst case of ESSAB to obtain the first feasible design uses 260 simulations, which is set as a threshold. Comparing ESSAB-GP and BO-wEI, the advantage of the new PFI infill sampling criterion compared with wEI can be observed when handling the complete set of specifications.

2.5.2 Case Study 2: ISA

The second test case is the inverter-stacking amplifier (ISA) (Figure 2.6) (72), which is a state-of-the-art structure. Although specifications based on AC analysis are extensive and promising, some transient analysis-based speci-

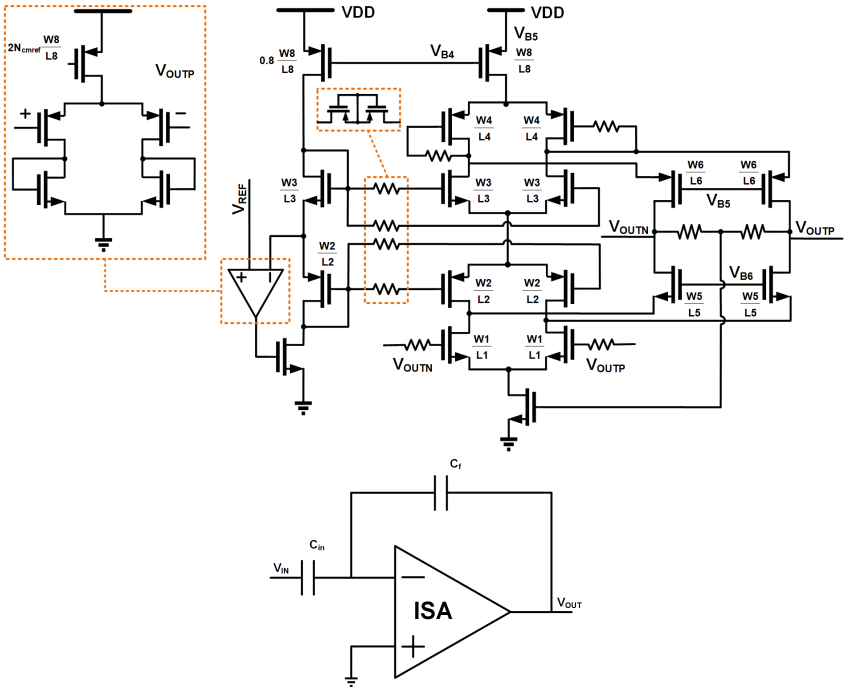


Figure 2.6: Schematic of the ISA

fications, which are often not easy to learn, are not used in this example as in (72). The purpose of selecting this example is to observe the behavior of different algorithms for such test cases compared to other more challenging test cases. This test case has 22 design variables, and the search ranges provided by the designer are in Table 2.5.

The sizing problem is defined as in Equation 2.10, which has 21 specifications in total.

Table 2.5: Design parameters and their ranges for ISA (test case 2)

Parameter	LB	UB	Parameter	LB	UB
L1(μm)	0.3	10	W3(μm)	0.22	40
L2(μm)	0.3	10	W4(μm)	0.22	40
L3(μm)	0.3	10	W5(μm)	0.22	40
L4(μm)	0.3	10	W6(μm)	0.22	40
L5(μm)	0.3	10	W7(μm)	0.22	40
L6(μm)	0.3	10	W8(μm)	0.28	40
L7(μm)	0.3	10	WR(μm)	0.4	40
L8(μm)	0.3	20	C_CMFB(fF)	10	30000
LR(μm)	0.3	10	Cf(fF)	10	30000
W1(μm)	0.22	40	C_in(fF)	10	30000
W2(μm)	0.22	40	Nmain (integer)	1	100

W: transistor width; L: transistor length; UB: upper bound; LB: lower bound

$$
\begin{aligned}
&\textbf{minimize}\ \text{Noise-Power Product} \\
&\textbf{s.t.}\quad \text{Open-loop Gain} \geq 70\ \text{dB} \\
&\qquad \text{DC-loop Gain} \geq 40\ \text{dB} \\
&\qquad \text{Closed-loop BW} \geq 30\ \text{kHz} \\
&\qquad \text{PMOS-input Degeneration Gain} \geq 30\ \text{dB} \\
&\qquad \text{NMOS-input Degeneration Gain} \geq 30\ \text{dB} \\
&\qquad \text{Output Offset 1-sigma} \leq 1 \times 10^{-3}\ \text{V} \\
&\qquad \text{Replica CMFB Loop Gain} \geq 13\ \text{dB} \\
&\qquad \text{Main CMFB Loop Gain} \geq 35\ \text{dB} \\
&\qquad \text{Differential Loop Phase Margin} \geq 65\ \text{deg} \\
&\qquad \text{Replica CMFB Loop Phase Margin} \geq 65\ \text{deg} \\
&\qquad \text{Main CMFB Loop Phase Margin} \geq 65\ \text{deg} \\
&\qquad \text{Closed-loop DC Gain} \geq 0\ \text{dB} \\
&\qquad \text{Vds Mismatch main/rep c.s.} \leq 0.1 \\
&\qquad \text{Output CM Voltage (max)} \leq 0.5\ \text{V} \\
&\qquad \text{Output CM Voltage (min)} \geq 0.4\ \text{V} \\
&\qquad \text{Saturation Margins} \geq 150\ \text{mV}
\end{aligned}
\tag{2.10}
$$

Ten runs are carried out for ESSAB and all of them successfully satisfy all the specifications. ESSAB converges within 200 simulations, which is fewer than the simulation budget (500 simulations). The total sizing time is about 1.8 hours (system time). In particular, all of them outperform the design in (72) (both based on simulation results). A typical design obtained by ESSAB is shown in Table 2.6 (including current biases) with the corresponding performance in Table 2.7.

Table 2.6: A typical design obtained by ESSAB (test case 2)

Parameter	Value	Parameter	Value	Parameter	Value
L1(μm)	1.7	W1(μm)	30.3	C_CMFB(fF)	160
L2(μm)	9.6	W2(μm)	0.8	Cf(fF)	860
L3(μm)	2.3	W3(μm)	0.9	C_in(fF)	13600
L4(μm)	8.2	W4(μm)	37.3	Nmain (integer)	18
L5(μm)	4.2	W5(μm)	4.2	Id_MB0 (nA)	288
L6(μm)	8.8	W6(μm)	2.4	Id_MB1 (nA)	360
L7(μm)	9.5	W7(μm)	13.2	Id_MBrep (nA)	20
L8(μm)	19	W8(μm)	1		
LR(μm)	6.1	WR(μm)	24.7		

The statistical results for all the reference methods and the PII curve are shown in Table 2.8 and Figure 2.7, respectively. It can be seen from Table 2.8 that all the reference methods successfully satisfy the specifications.

Table 2.7: Performance values of a typical design obtained by ESSAB (test case 2)

Noise-Power Product 1.81 $pWHz$	Open-loop Gain 70.0 dB	DC-loop Gain 46.5 dB
Closed-loop BW 33 kHz	P-input Deg. Gain 41.9 dB	N-input Deg. Gain 54.8 dB
Out. Offset 1-sigma 5.9×10^{-4} V	Rep. CMFB loop gain 14.1 dB	Main CMFB loop gain 60.0 dB
Diff. Loop PM 89 $deg.$	Rep. CMFB Loop PM 146 $deg.$	Main CMFB Loop PM 88 $deg.$
Closed-loop DC Gain 23.7 dB	Vds Mismatch main/rep 0.05	Output CM V. (max.) 0.41 V
Output CM V. (min.) 0.41 V	Saturation Margins all satisfied	

All surrogate model-based methods are much more efficient than DE. In Figure 2.7, the worst case of ESSAB to obtain the first feasible design uses 79 simulations, which is set as a threshold. It can be seen that the PII convergence trends are comparable for ESSAB and ESSAB-GP, despite that ESSAB is more efficient. BO-wEI is slightly slower, showing the advantage of the proposed PFI compared to wEI even without considering transient analysis-based specifications.

Besides, two more observations can be made: (1) Considering the complete set of specifications, although the total number is important, difficult-

Table 2.8: Statistical results for different algorithms (test case 2)

Algorithm	DE	BO-wEI	ESSAB-GP	ESSAB
Success rate	10/10	10/10	10/10	10/10
N_{feasible}	1300	100	73	**50**
Min. noise-power product ($pWHz$)	2.12	**1.68**	1.96	1.72
Max. noise-power product ($pWHz$)	2.55	2.6	2.48	**1.96**
Mean noise-power product ($pWHz$)	2.37	2.27	2.22	**1.81**
Std. noise-power product ($pWHz$)	0.17	0.25	0.22	0.13
Modeling time (h)	N.A.	28	6.5	0.5
Simulation time (h)	72	3.6	3.6	3.6

to-learn performances play a critical role. The reason is that providing an accurate prediction for them is more difficult due to their highly nonlinear characteristics and the unavoidable, many simulation failures. When removing them, such as in this test case, all surrogate model-based methods succeed. (2) Even under this condition, ESSAB is better than all other reference methods. In Table 2.8, the average objective function value of ESSAB is about 20% better than DE, BO-wEI and ESSAB-GP. ESSAB also has the best efficiency reflected by the small number of simulations as well as the short modeling time.

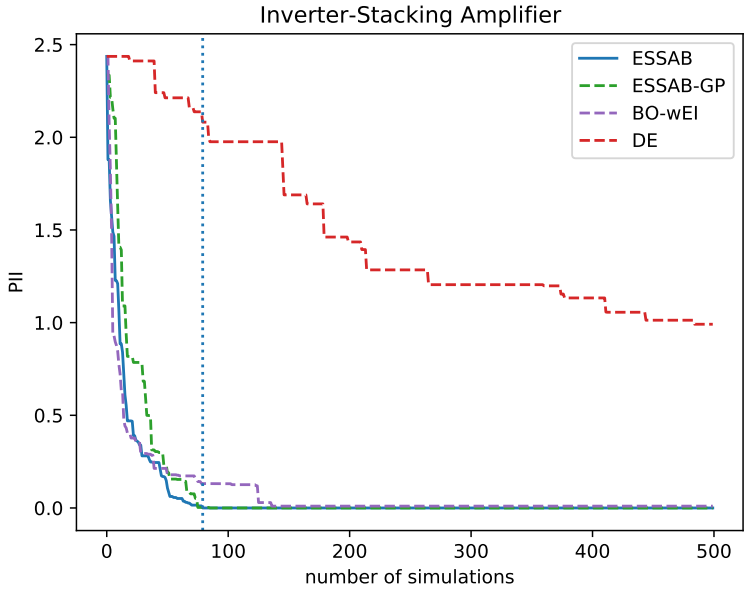


Figure 2.7: ISA PII Values (average over 10 runs)

2.5.3 Case Study 3: SA Comparator

The third test case is a strong-arm latch comparator which is shown in Fig. 2.8. It has 13 design variables and the search ranges provided by the designer are in Table 2.9. Following (81), the specifications are stringent.

The sizing problem is defined as in Equation 2.11, which has 10 speci-

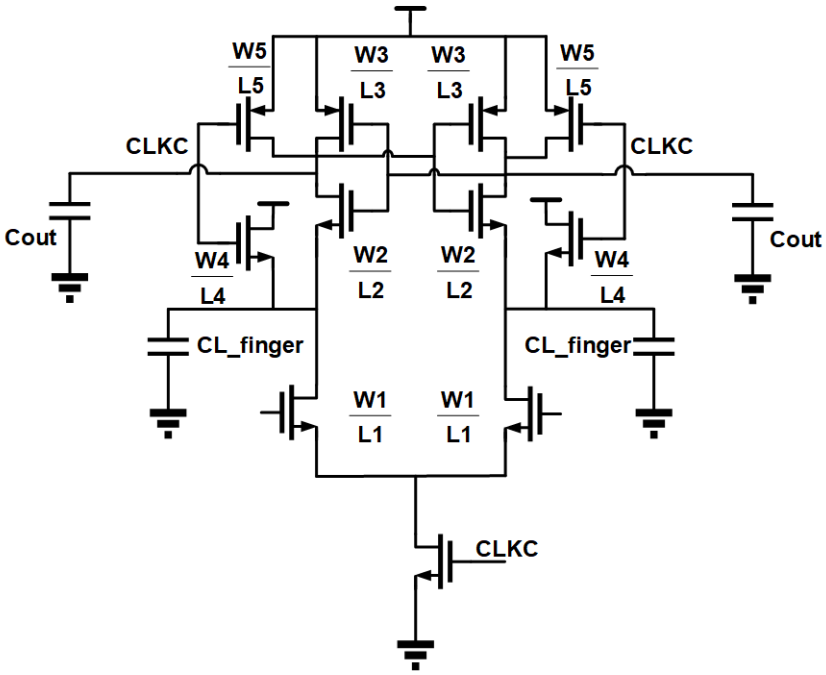


Figure 2.8: Schematic of the SA Comparator

fications.

$$
\begin{aligned}
&\textbf{minimize } \text{Power} \\
&\textbf{s.t.} \quad \text{Set Delay Time} \leq 10 \ \text{ns} \\
&\qquad \text{Reset Delay Time} \leq 6.5 \ \text{ns} \\
&\qquad \text{Area} \leq 26 \ \mu\text{m}^2 \\
&\qquad \text{Input-ref Noise} \leq 5 \times 10^{-5} \ \text{Vrms} \\
&\qquad \text{Differential Reset Voltage} \leq 1 \ \mu\text{V} \\
&\qquad \text{Differential Set Voltage} \geq 1.195 \ \text{V} \\
&\qquad \text{Positive Integration Node Reset Voltage} \leq 60 \ \mu\text{V} \\
&\qquad \text{Negative Integration Node Reset Voltage} \leq 60 \ \mu\text{V} \\
&\qquad \text{Positive Output Node Reset Voltage} \leq 0.35 \ \mu\text{V} \\
&\qquad \text{Negative Output Node Reset Voltage} \leq 0.35 \ \mu\text{V}
\end{aligned}
\tag{2.11}
$$

Table 2.9: Design parameters and their ranges for SA Comprator

Parameter	LB	UB	Parameter	LB	UB
L1(μm)	0.18	10	W1(μm)	0.22	50
L2(μm)	0.18	10	W2(μm)	0.22	50
L3(μm)	0.18	10	W3(μm)	0.22	50
L4(μm)	0.18	10	W4(μm)	0.22	50
L5(μm)	0.18	10	W5(μm)	0.22	50
L6(μm)	0.18	10	W6(μm)	0.28	50
Cl_finger (integer)	10	300			

W: transistor width; L: transistor length; UB: upper bound; LB: lower bound

Ten runs are carried out for ESSAB and all of them successfully satisfy all the specifications. ESSAB converges within the simulation budget (500 simulations). The total sizing time is about 3.8 hours (system time). In particular, all of them outperform the design in (81) (both based on simulation results). A typical design obtained by ESSAB is shown in Table 2.10 with the corresponding performance in Table 2.11.

The statistical results for all the reference methods and the PII curve are shown in Table 2.12 and Fig. 2.9, respectively.

With the complete set of specifications, ESSAB outperforms other reference methods considering both solution quality and efficiency. Similar con-

Table 2.10: A typical design obtained by ESSAB (test case 3)

Parameter	Value	Parameter	Value	Parameter	Value
L1(μm)	0.18	L6(μm)	0.18	W5(μm)	3.5
L2(μm)	0.18	W1(μm)	50.0	W6(μm)	4.3
L3(μm)	0.18	W2(μm)	5.0	Cl_finger (integer)	44
L4(μm)	0.18	W3(μm)	5.2		
L5(μm)	0.18	W4(μm)	4.7		

Table 2.11: Performance values of a typical design obtained by ESSAB (test case 3)

Power $2.7\mu W$	Set Delay Time $9.7\ ns$	Reset Delay Time $4.2\ ns$
Area $25.8\mu m^2$	Input-Ref Noise $4.9 \times 10^{-5} V_{rms}$	Diff. Reset Voltage $2.8 \times 10^{-7} \mu V$
Diff. Set Voltage $1.2V$	Pos-Integ. Res. V. $56\mu V$	Neg-Integ. Res. V. $57\mu V$
Pos-Output Res. V. $3 \times 10^{-4} \mu V$	Neg-Output Res. V. $3 \times 10^{-2} \mu V$	

clusions as test case 1 can be drawn, which will not be repeated. Additional observations include: (1) Within 500 simulations, ESSAB-GP only obtains feasible designs 6 times. Although separate runs using 1000 simulations show that ESSAB-GP can also obtain 100% success, it is much slower than ES-

Table 2.12: Statistical results for different algorithms (test case 3)

Algorithm	DE	BO-wEI	ESSAB-GP	ESSAB
Success rate	5/10	0/10	6/10	**10/10**
N_{feasible}	10000	N.A.	500	**390**
Min. power (μW)	2.98	N.A.	3.05	**2.52**
Max. power (μW)	4.22	N.A.	3.75	**2.73**
Mean power (μW)	3.57	N.A.	3.45	**2.66**
Std. power (μW)	0.5	N.A.	0.36	0.098
Modeling time (h)	N.A.	17	3	0.2
Simulation time (h)	70	3.5	3.5	3.5

SAB. (2) With such stringent specifications, mainly from transient and noise analysis-based performances, even DE does not obtain 100%.

2.5.4 Case Study 4: VCO

Another kind of typical analog building block is the Voltage Controlled Oscillator (VCO). Recently, some novel structures have been proposed (58; 82). Hence, the fourth test case is a distributed-input VCO (58) implemented in a 40 nm technology (Figure 2.10). It has 10 design variables, and the search ranges provided by the designer are in Table 2.13. Besides studying ESSAB's effectiveness for various building blocks, this test case is selected because: (1) Each simulation costs more than 10 minutes, and traditional global optimization methods may cost prohibitive time. (2) Although only having 5

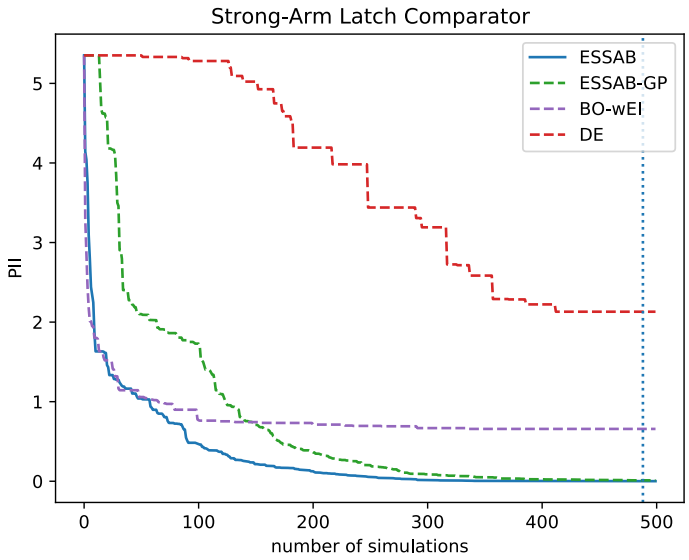


Figure 2.9: SA Latch Comparator PII Values (average over 10 runs)

specifications, the specifications are stringent, and the frequency/voltage gain (KVCO) is difficult to learn due to the high simulation failure rate.

Table 2.13: Design parameters and their ranges for VCO (test case 4)

Parameters	LB	UB	Parameters	LB	UB
Idc(μA)	10	75	Ltail(μm)	0.04	20
Lctrl(μm)	0.04	1	Ln(μm)	0.04	0.4
Lp(μm)	0.04	0.4	Wtail(μm)	0.12	100
Wctrl(μm)	0.12	100	Wn(μm)	0.12	100
Wp(μm)	0.12	100	Current ratio (integer)	1	20

W: transistor width; L: transistor length; UB: upper bound; LB: lower bound

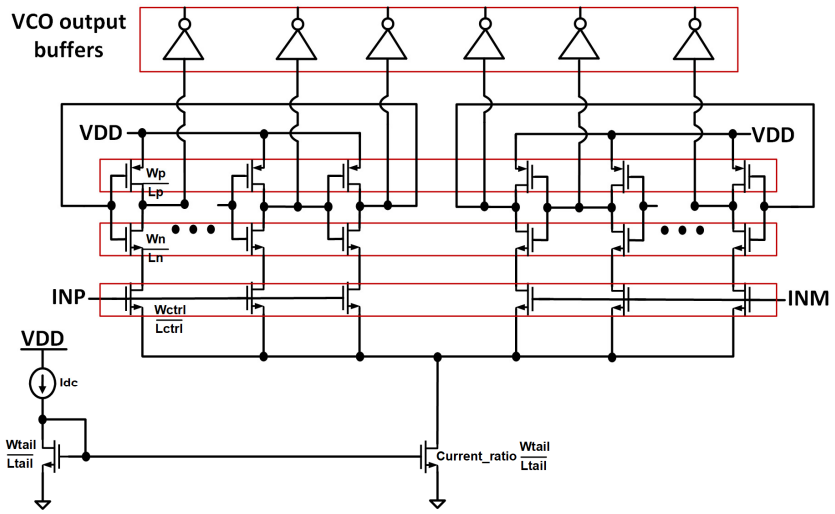


Figure 2.10: Schematic of VCO

The sizing problem is defined as follows, which has 5 specifications.

$$\begin{aligned} &\textbf{minimize} \text{ Noise-Power Product} \\ &\textbf{s.t.} \quad \text{Center Frequency} \geq 75 \text{ MHz} \\ &\qquad \text{KVCO min} \geq 1.28 \text{ GHz/V} \\ &\qquad \text{KVCO max} \leq 1.42 \text{ GHz/V} \\ &\qquad \text{Area} \leq 300 \ \mu\text{m}^2 \\ &\qquad \text{Power} \leq 150 \times 10^{-6} \text{ W} \end{aligned} \tag{2.12}$$

Ten runs are carried out for ESSAB, and they satisfy all the specifications. ESSAB converges within the simulation budget (200 simulations). The total sizing time is about 25 hours (system time). In particular, they all outperform the design in (58) (both based on simulation results). A typical design obtained by ESSAB is shown in Table 2.14 with the corresponding performance in Table 2.15.

Table 2.14: A typical design obtained by ESSAB (test case 4)

Parameter	Value	Parameter	Value	Parameter	Value
Idc(μA)	23.7	Ltail(μm)	2.45	Lctrl(μm)	0.11
Ln(μm)	0.08	Lp(μm)	0.07	Wtail(μm)	0.64
Wctrl(μm)	6.60	Wn(μm)	0.42	Wp(μm)	1.82
Current ratio	1				

Table 2.15: Performance values of a typical design obtained by ESSAB (test case 4)

Noise-Power Product	Center Frequency	Area
$0.75fWHz$	$90.9MHz$	$260\mu m^2$
Power	KVCO	
$65\mu W$	$1.32GHz/V$	

Table 2.16 and Fig. 2.11 show the statistical results for all the reference methods and the PII curve, respectively. As mentioned above, DE is expected to cost prohibitive time and is not used for this test case.

ESSAB outperforms other reference methods considering both solution quality and efficiency. The same conclusions as test case 1 can be drawn, which will not be repeated. A new observation is that after 200 simulations, the noise-power product of ESSAB (on average) is about 3 times better than that of ESSAB-GP. However, ESSAB-GP can obtain a similar value after 400 simulations. This shows ESSAB's advantage in efficiency even clearer.

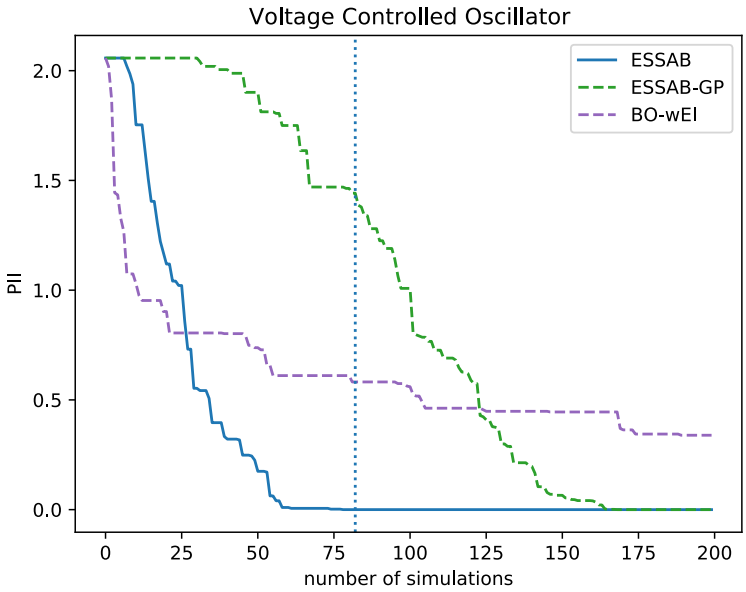


Figure 2.11: VCO PII Values (average over 10 runs)

Table 2.16: Statistical results for different algorithms (test case 4)

Algorithm	BO-wEI	ESSAB-GP	ESSAB
Success rate	3/10	10/10	**10/10**
N_{feasible}	N.A.	141	**54**
Min. noise-power product ($10^{-16} \times WHz$)	10.3	6.3	**5.5**
Max. noise-power product ($10^{-16} \times WHz$)	57.2	100	**10.6**
Mean. noise-power product ($10^{-16} \times WHz$)	33.2	28.8	**8.2**
Std. noise-power product ($10^{-16} \times WHz$)	23.4	35.4	1.9
Modeling time (h)	1.1	0.2	0.05
Simulation time (h)	25	25	25

2.6 Summary

This chapter proposes an ML-based modeling technique to improve the efficiency of global optimization methods for AMS circuit sizing. We utilized Artificial Neural Networks to approximate design performance without costly SPICE simulations. Further, we addressed common issues such as the modeling under data scarcity and parameter-free infill sampling methodology. Compared to conventional counterparts, the resulting algorithm used a significantly lower number of simulations to optimize the design parameters of AMS circuits.

Chapter 3: DNN-Opt: An RL Inspired Optimization for Analog Circuit Sizing using Deep Neural Networks[1]

This chapter introduces a Reinforcement Learning inspired solution to the analog sizing problem. We show how to represent state, action, and return values to solve a continuous space design exploration problem and optimize analog performance metrics. Further, a recipe is formulated to extend this method into industrial-scale circuits with very large design variables. The effectiveness of the introduced method is demonstrated on both academic and advanced node industrial circuits.

3.1 Introduction

Recently, reinforcement learning algorithms are applied in the area as learning-based methods. GCN-RL(86) leverages Graph Neural Networks (GNN) and proposes a transferable framework. Despite reporting superior results over various methods and human-designer, a) it requires thousands of

[1]This chapter is based on the following publication: Ahmet Faruk Budak, Prateek Bhansali, Bo Liu, Nan Sun, David Z Pan and Chandramouli V Kashyap "DNN-Opt: An RL Inspired Optimization for Analog Circuit Sizing using Deep Neural Networks," in ACM/IEEE Design Automation Conference (DAC), 2021. I am the main contributor in charge of problem formulation, algorithm development and experimental validations.

simulations for convergence (without transfer learning) and b) it suffers from engineering effort to determine observation vector, architecture selection, and reward engineering. AutoCkt (69) is a sparse sub-sampling RL technique optimizing the circuit parameters by taking discrete actions in the solution space. AutoCkt shows more efficiency over random RL agents and Differential Evolution. Still, it requires to be trained with thousands of SPICE simulations before deployment, which is costly.

The rest of this chapter introduces DNN-Opt, a two-stage deep learning black-box optimization scheme, where we merge the strengths of Reinforcement Learning (RL), Bayesian Optimization (BO), and population-based techniques in a novel way. The key features of the DNN-Opt framework are below.

- We tailored a two-stage Deep Neural Network (DNN) architecture for black-box optimization tasks inspired by the actor-critic algorithms developed in the RL community.
- To leverage the convergence behavior of population-based methods, DNN-Opt adopts a population-based search space control mechanism.
- We introduce a recipe for extending our work for large industrial designs using sensitivity analysis. In collaboration with a design house,

we demonstrate that our work can also efficiently size large circuits with tens of thousands of devices in addition to small building blocks.

3.2 Background

In this section, we will provide a brief definition for reinforcement learning (RL) and its use in Electronic Design Automation (EDA).

3.2.1 Reinforcement Learning in EDA

Reinforcement learning (RL) is a field of machine learning that studies the relationship between the agents and the environment they interact with. In general, an RL agent's objective is to maximize the accumulated reward due to its interaction with the environment. It has shown huge success and even demonstrated superhuman performance in well-defined environments, such as playing video games (71), chess, and Go (75). In addition, RL algorithms are applied in many real-world applications, where environments are much more complicated.

Recently, the performance of RL algorithms are boosted by the representation ability of deep learning models, creating a new subfield of Deep reinforcement learning. It especially has been proven to be effective in environments where the environment representation, i.e., state, is high dimensional. Many effective Deep RL algorithms have been developed both for discrete and

continuous action space.

Many EDA problems are considered as high-dimensional optimization problems and are interest of Deep RL algorithms. Therefore, we recently observed a high focus on bringing RL-based solutions to EDA problems both in industry and academia (7; 68; 64; 31). In academia, we have seen RL algorithms used to tackle many sub-problems in EDA, including placement, sizing, logic synthesis, routing, etc. In industry, we notice that companies in EDA have devoted much effort in the direction of general artificial intelligence (AI). For instance, Synopsys and Cadence have developed and promoted their AI platforms DSO.ai [2] and Cerebrus [3].

RL algorithms solves the problems by first modeling their environment into a Markov decision process (MDP). A Markov decision process consists of 4 main elements, (S, A, P, R), and they are defined as follows:

- S and A are state and action spaces, respectively.
- P is the state transition function, $P(s, s', a) = \texttt{Pr}(S_{t+1} = s' | S_t = s, A_t = a)$ is the probability that action a in state s will result in state s' at next timestamp.

[2]Link to Synopsys DSO.ai
[3]Link to Cadence Cerebrus Intelligent Chip Explorer

- R is the reward function.

At the t-th step of a Markov decision process, we observe the state S_t and reward R_t from the environment and decide the action $A_t = \pi(s_t)$ based on the policy $\pi(s_t)$. We repeat the process of observation and decision by interacting with the environment. Our objective is to find the optimal policy function π^* to maximize the expectation of accumulated reward formulated as follows,

$$\mathbb{E}[\sum_{t=0}^{T} \gamma^t R_t]$$

where $\gamma \in [0, 1]$ is the discounting factor and T is the total number of steps of one episode. $T = \infty$ means that the MDP continues without termination.

Reinforcement learning algorithms can solve this control problem by learning from repeated trial and error. There are two major branches in reinforcement learning algorithms: value-based and policy-based methods. In value-based methods, we attempt to model and calculate the value of each state $s \in S$ or each pair of state and action $(s, a) \in (S, A)$. In policy-based methods, we parameterize the policy function π_θ and optimize the parameters θ. We refer the readers to (80) for more details.

3.3 Methodology

DNN-Opt follows a typical flow of AMS sizing algorithms. It is first initialized randomly, and the core algorithm works in an iterative way to guide

the search for optimal solutions. Our innovations in this work are to propose an efficient RL-based core algorithm and a sensitivity analysis-based input space reduction to extend this method to large-scale applications.

3.3.1 Core DNN-Opt Algorithm

The overall framework of DNN-Opt is shown in Figure 3.1. DNN-Opt comprises a two-stage deep neural network architecture that interacts with a circuit simulator during the optimization process. The flow starts from generated samples in the design space; then, a critic-network is used to predict any new design point's performance. This prediction is used by the actor network to propose new candidates for simulation. This search scheme efficiently mimics BO behavior in space exploration. Besides, the sample generation is further optimized by adopting a population control scheme.

The two-stage network architecture of our work borrows its structure from Deep Deterministic Policy Gradient (DDPG) algorithm (40), which is an RL actor-critic algorithm (36) developed for continuous action spaces. However, actor-critic algorithms are not directly applicable to analog circuit sizing since it is not a Markov Decision Process (MDP) (80), which is a *necessary condition* for any RL problem. Therefore we adapt the DDPG algorithm with significant modifications tailored for analog circuit sizing.

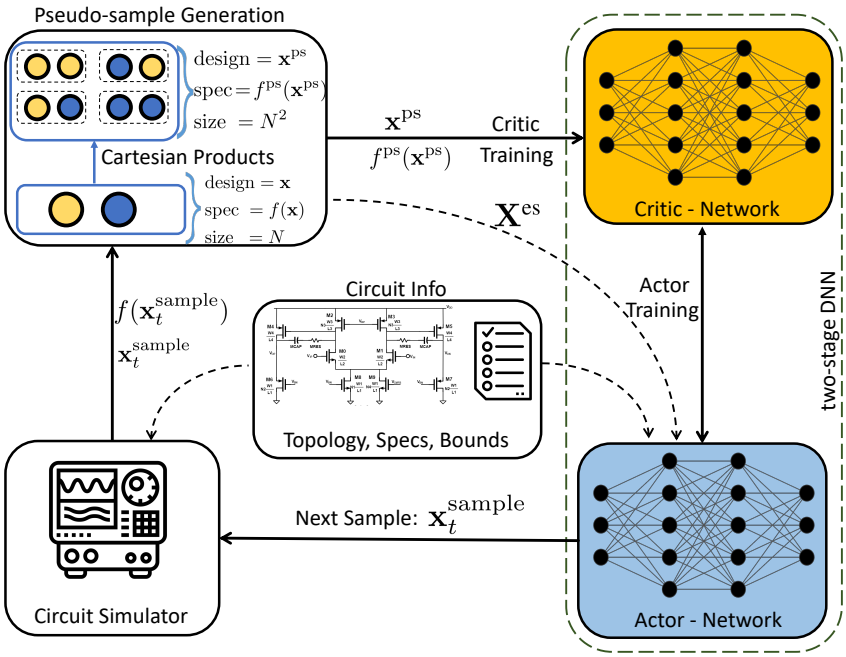


Figure 3.1: DNN-Opt Framework

3.3.2 Modeling Sizing Problem as RL Environment

In the context of analog circuit sizing, we will keep some of the RL notation but replace many for simplicity and clarity.

Design: A design is a set of circuit parameters that we denote by $\mathbf{x}$ and is a vector of size d where each element corresponds to a particular design variable. The optimization goal is to find optimal $\mathbf{x}_{\text{opt}}$, which satisfies Equation 2.1.

Population: A population is a set of multiple designs.

Design Population Matrix: We define a design population matrix as $\mathbf{X} \in$

$\mathbb{R}^{N\times d}$, where N is the population size. The parameters of i^{th} design is a row in the design population matrix $\mathbf{X}$, which is denoted as $\mathbf{x}_i$.

State Space: Our work maps optimization parameters (circuit design variables) to state representation in RL notation. A state of k^{th} design is transformed as $\mathbf{s}_k = \mathbf{x}_k$.

Action Space: Each action $\mathbf{a}_k$ in our new architecture corresponds to *change* in optimization parameters vector, $\mathbf{x}_k$, which can be denoted as $\mathbf{a}_k = \Delta\mathbf{x}_k$. An intuitive explanation of this choice is that an ideal action for an optimization task should propose a change in each design variable to have a better design.

Critic-Network: Originally, a critic-network parameterized by θ^Q approximates the return value of an MDP Return $= Q(s_t, a_t|\theta^Q)$. We modify its role and use this network as a proxy in lieu of expensive SPICE simulator. Our modified critic-network provides a vector-to-vector mapping by taking an $(\mathbf{x}, \Delta\mathbf{x}) \in \mathbb{D}^{2d}$ as input and providing performance predictions $Q(\mathbf{x}, \Delta\mathbf{x}|\theta^Q) \in \mathbb{R}^{m+1}$ at output, one-dimension is for objective specification and m for constraint specifications.

Actor-Network: An actor-network parameterized by θ^μ would take a state as its input and determine an action to take $\mathbf{a}_k = \mu(\mathbf{s}_k|\theta^\mu)$. In the context of analog circuit sizing, actor-network provides change in design parameter vector for design k as: $\Delta\mathbf{x}_k = \mathbf{a}_k = \mu(\mathbf{x}_k|\theta^\mu)$.

Critic-Network Training: We utilize critic-network for modeling design

variable to circuit performance relationship. For effective training, we use data augmentation techniques to generate N^2 *pseudo-samples* (ps) using original N samples. In order to generate pseudo-samples, we use two-samples $\mathbf{x}_i$ and $\mathbf{x}_j$ and corresponding spec vectors $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$, as follows:

$$\begin{aligned} \mathbf{x}_{ij}^{\text{ps}} &= [\mathbf{x}_i, \Delta\mathbf{x}_{ij}] = [\mathbf{x}_i, \mathbf{x}_j - \mathbf{x}_i] \\ f^{\text{ps}}(\mathbf{x}_{ij}^{\text{ps}}) &= f(\mathbf{x}_j) \end{aligned} \tag{3.1}$$

This leads to a change in the input dimensionality of critic-network from d to $2d$ since we now have to use $(\mathbf{x}, \Delta\mathbf{x})$ instead of $\mathbf{x}$ or $(\mathbf{x}+\Delta\mathbf{x})$. Our experiments conducted on Bayesmark (84) benchmark problems showed that using $2d$ inputs and training with pseudo-samples boosted critic-network's accuracy significantly over a network trained with d inputs and original samples.

For a batch size of N_b pseudo-samples, the following Mean Squared Error (MSE) loss function is used to train the critic network.

$$\mathrm{L}\big(\theta^Q\big) = \tfrac{1}{N_b(m+1)} \sum_{k=1}^{N_b} \sum_{l=1}^{m+1} \big(Q(\mathbf{x}_k, \Delta\mathbf{x}_k)^l - f(\mathbf{x}_k + \Delta\mathbf{x}_k)^l\big)^2 \tag{3.2}$$

where $Q(\mathbf{x}_k, \Delta\mathbf{x}_k)^l$ is the critic-network's approximation for kth pseudo-sample's l^{th} performance and $f(\mathbf{x}_k + \Delta\mathbf{x}_k)^l$ is the SPICE simulated value for the same design-performance pair. To clarify, we have SPICE simulation values for pseudo-samples because of the way they are constructed.

Actor-Network Training: Training of the actor-network is done after the critic-network is trained and its hyperparameters are fixed. The training of

actor-network corresponds to search in design space for *better* designs. We develop a Figure of Merit (FoM) function, $g(\cdot)$, based on performance-vector to objectively quantify how better a design is with respect to others.

$$g\left[f(\mathbf{x})\right] = w_0 \times f_0(\mathbf{x}) + \sum_{i=1}^{m} \min\left(1, \max(0, w_i \times f_i(\mathbf{x}))\right) \tag{3.3}$$

where w_i is the weighting factor. Note, a $\max(\cdot)$ clipping is used for equating designs after constraints are met, and $\min(\cdot)$ clipping is used for practical purposes to prevent single constraint violation to dominate $g(\cdot)$ value. We train actor-network parameters by using $g(\cdot)$ function and replacing SPICE simulation values $f(\cdot)$ by the critic-network predictions $Q(\mathbf{x}, \Delta\mathbf{x})$. We will further use a population of "elite" solutions (es) of size N_{es} to restrict search space for the actor-network. The population of elite solutions is a subset of the total population determined based on the FoM ranking.

For a batch size of N_b samples, the following loss function is used to train actor-network.

$$L\left(\theta^{\mu}\right) = \frac{1}{N_b} \sum_{k=1}^{N_b} \left(g\left[Q(\mathbf{x}_k, \mu(\mathbf{x}_k \mid \theta^{\mu}))\right] + \|\lambda * \text{viol}_k\|_2\right) \tag{3.4}$$

where $\mu(\mathbf{x}_k \mid \theta^{\mu})$ is proposed parameter change vector $\Delta\mathbf{x}_k$ by the actor network. $(\lambda * \text{viol}_k)$ is an element-wise vector multiplication where λ is the weighting coefficient chosen to be very large to prevent any boundary violation and keep the search in the restricted search region. The total boundary violation

viol_k for action k is defined as follows:

$$\text{viol}_k = \max(0, lb_{\text{rest}} - (\mathbf{x}_k + \Delta\mathbf{x}_k)) + \max(0, (\mathbf{x}_k + \Delta\mathbf{x}_k) - ub_{\text{rest}}) \quad (3.5)$$

where lb_{rest} and ub_{rest} are the restriction boundary vectors for design variables determined by the population of elite solutions given by:

$$lb^i_{\text{rest}} = \min(\mathbf{x}^i) \;\; \forall i = 1, \ldots, d$$

$$ub^i_{\text{rest}} = \max(\mathbf{x}^i) \;\; \forall i = 1, \ldots, d$$

where, $\mathbf{x}^i$ is the column vector of size N_{es} consisting of i^{th} parameter of all designs in the elite population.

The hyperparameters (number of layers, number of nodes, learning rate, etc.) of the architecture for the actor and critic networks were found based on empirical studies.

3.3.3 Recipe for Industrial Circuits

We use sensitivity analysis to prune design search space for efficiently finding an optimized solution. A blind search space exploration may lead to wasted circuit simulations during optimization. For example, in a classical seven transistor Operational Amplifier (OpAmp) (17), power dissipation does not depend on the differential pair devices once they are in saturation. Thus, if we want to size a circuit to reduce power, we should not make device properties of the differential pair devices as variables. To use sensitivity analysis in

practice for any generic circuit, we first traverse the circuit hierarchy and collect all unique device design variables, d. Then, we perform sensitivity analysis by perturbing each design variable around its nominal value and observing its impact on objective and constraints, f_i. More formally, we compute sensitivity $\mathcal{S}_{ij}$ as

$$\mathcal{S}_{ij} = \frac{\delta f_i}{\delta d_j}, \forall i = 0, \ldots, m; j = 1, \ldots, d. \tag{3.6}$$

We only need to consider design variables for which $\mathcal{S}_{ij} > thresh$, where $thresh$ is a user-defined number. Empirically, this analysis prunes design search space effectively, allowing us to work on large-scale circuits.

We are now ready to present the overall framework of DNN-Opt in the following subsection.

3.3.4 Overall Framework

The overall framework for DNN-Opt is provided in Algorithm 2. As a prerequisite, we apply sensitivity analysis for a large design and reduce the number of design variables to a workable range. We then randomly sample N_{init} points from the design search space to build the initial population. For optimization iteration t, first step is to initialize actor-critic parameters followed by pseudo-sample generation. Next, actor-network and critic-network

Algorithm 2 DNN-Opt Algorithm

Require: Dimensionality reduction with sensitivity analysis **if** design is *large*
Require: An initial sample set $\mathbf{X}^{\text{init}}$ of N_{init} designs and their evaluations $f(\mathbf{X}^{\text{init}})$

1: Define total population $\mathbf{X}^{\text{tot}} = \mathbf{X}^{\text{init}}$
2: **for** $t = 1, 2, \ldots, t_{max}$ **do**
3: Initialize actor & critic network parameters θ^{μ} and θ^{Q}
4: Generate pseudo-samples using existing design $\mathbf{X}^{\text{tot}} \rightarrow$ Eqn. 3.1
5: Train critic-network $\rightarrow$ Eqn. 3.2
6: Train actor-network $\rightarrow$ Eqn. 3.4
7: Calculate FoM for each design by FoM $= g[f(\mathbf{X}^{\text{tot}})]$
8: Choose N_{es} designs with smallest FoM to form population of elite solutions $\mathbf{X}^{\text{es}}$.
9: Find query point (next sample) $\mathbf{x}_t^{\text{sample}}$ using actor-model $\rightarrow$ Eqn. 3.7
10: Simulate the query point and obtain specs $f(\mathbf{x}_t^{\text{sample}})$ via SPICE sims
11: **if** return cond(e.g. specs are met) **then**
12: break
13: **end if**
14: $\mathbf{X}^{\text{tot}}$.append($\mathbf{x}_t^{\text{sample}}$)
15: Go back to line 3
16: **end for**
17: **return** The design with the highest FoM

are trained. After this, an elite population is constructed based on the FoM of total population (this elite population will be updated with optimization iterations). The next query point is generated from elite-population, $\mathbf{X}^{\text{es}}$, using a pre-trained actor-critic as follows. We use every design, $\mathbf{x}_i^{\text{es}}$, in the pool of elite-population as input to the actor-network. The output of actor-network, $\Delta\mathbf{x}_i^{\text{es}} = \mu(\mathbf{x}_i^{\text{es}})$, is a proposed change for design parameters in search of an

optimal solution. With the imposed exploration noise ($\mathcal{N}$), a candidate design point is naturally formed as: $\mathbf{x}_i^{\text{ca}} = \mathbf{x}_i^{\text{es}} + \mu(\mathbf{x}_i^{es}) + \mathcal{N}$. At this step, we have exactly the same number of proposed candidates, $\mathbf{X}^{\text{ca}} = [\mathbf{x}_i^{\text{ca}}, \ldots, \mathbf{x}_{N_{\text{es}}}^{\text{ca}}]$, as the size of elite-population. Once the population pairs, $\mathbf{X}^{\text{es}}$ and $\mathbf{X}^{\text{ca}}$ are formed, the next sample point for iteration t is selected using Equation 3.7.

$$\mathrm{x}_t^{\text{sample}} = \left[\mathbf{x}_k^{\text{ca}} \text{ for } k = arg\,min_i \left(g[Q(\mathbf{x}_i^{\text{es}}, \mathbf{x}_i^{\text{ca}} - \mathbf{x}_i^{\text{es}})]\right)\right] \tag{3.7}$$

3.4 Experiments

To demonstrate the reliability and efficiency of the DNN-Opt, we apply it to two sets of experiments using six circuit examples. The first experiment set is on small building blocks where every transistor is parameterized and sized, and the second experiment set includes larger industrial circuits with thousands of nodes and devices.

3.4.1 Experiments on Academic Building Blocks

We tested DNN-Opt on two small building blocks: a folded cascode amplifier and a strong-arm latch comparator. We included the majority of the circuit performances in the constraint list to mimic real-world design experience. Both designs are implemented in 180nm CMOS technology.

We compare our algorithm with three other well-known methods: a)

A Differential Evolution (DE) method, which is a conventional population-based model-free algorithm, b) Bayesian Optimization with weighted Expected Improvement (BO-wEI)(57), which is a modified version of Bayesian Optimization for constrained problems, and c) GASPAD method(41), a surrogate model (GP) assisted evolutionary framework. To account for the randomized techniques involved in all these methods, we repeat experiments ten times to report each method's findings. We determine the simulation budgets for our experiments by considering the convergence nature of the methods. DE has a simulation budget of 10000, and BO-wEI, GASPAD, and DNN-Opt are limited by 500 simulations. All the experiments are run on a workstation with Intel Xeon CPU and 128GB RAM, and a commercial SPICE simulator. We used several metrics to compare the algorithms. We provide statistics of the methods for each example, and we denote the number of times a feasible solution is found by *success rate*. We also share the evolution of FoM value calculated based on Equation 2.2 to demonstrate each algorithm's convergence during runtime. The constraint expressions given in Equation 3.8 and 3.9 can be trivially readjusted to fit into the form of Equation 2.1.

Folded Cascode OTA: The first test case is a two-stage folded-cascode Operational Transconductance Amplifier (OTA) (Figure 2.4).It has 20 design variables, and the designer provided search ranges are as shown in Table 3.1.

Table 3.1: Folded-Cascode OTA Search Space

Parameter Name	Unit	LB	UB
L1-L2-L3-L4-L5-L6-L7	μm	0.18	2
W1-W2-W3-W4-W5-W6-W7	μm	0.24	150
N1-N2-N8-N9	integer	1	20
MCAP	fF	100	2000
Cf	fF	100	10000

W:device width; L:device length; UB:upper bound; LB:lower bound

The sizing problem is defined as follows:

$$\begin{aligned} &\text{minimize Power} \\ &\text{s.t.} \quad \begin{array}{ll} \text{DC Gain} > 60 \text{ dB} & \text{Settling Time} < 30 \text{ ns} \\ \text{CMRR} > 80 \text{ dB} & \text{Saturation Margin} > 50 \text{ mV} \\ \text{PSRR} > 80 \text{ dB} & \text{Unity Gain Freq.} > 30 \text{ MHz} \\ \text{Out. Swing} > 2.4 \text{ V} & \text{Out. Noise} < 30 \text{ mV}_{\text{rms}} \\ \text{Static error} < 0.1 & \text{Phase Margin} > 60 \text{ deg.} \end{array} \end{aligned} \tag{3.8}$$

In our experiment, the following transistors are required to operate in the saturation region: M1, M3, M4, M7, M9, M10, M12, M13, and [M15-M26]. The total number of design constraints becomes 29.

The statistical results for all the reference algorithms are shown in Table 3.2. DNN-Opt shows high reliability and find a feasible solution in all its trials. However, other model-based methods, BO-wEI and GASPAD, fail to achieve similar behavior. DE can also find feasible results, but DNN-Opt is 24x more efficient in the number of required simulations to find the first feasible result. It is also demonstrated in Table 3.2 that, on average, the final design proposed by DNN-Opt draws up to 43% less power. The modeling time required by DNN-Opt is up to 50x smaller compared to other model-based methods. This results in 2.5–16x efficiency for total runtime.

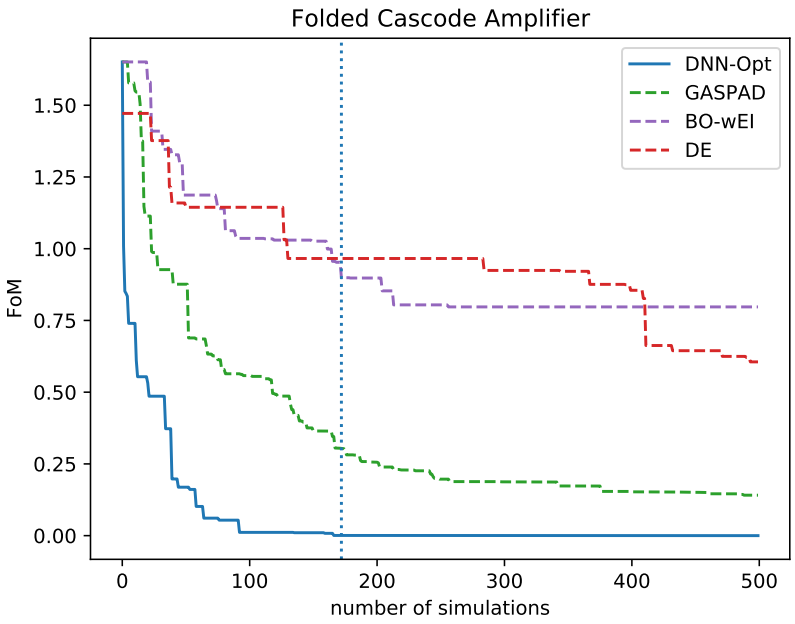


Figure 3.2: Folded Cascode OTA FoM Convergence

Figure 3.2 includes the FoM curve with iterations, where DNN-Opt shows strong convergence behavior and outperforms other methods. For our ten runs, DNN-Opt finds the feasible solution within 205 iterations (marked with vertical dashed line) across all its ten trials. Although it is slow, GASPAD shows convergence to optimal FoM, but we observed that BO-wEI is often trapped in local optima.

Strong-Arm Latch Comparator: The second test case is SA-Latch Comparator, which is shown in Figure 2.8. It has 13 design variables, and their names and bounds are shown in table 3.3.

The constrained optimization problem consists of 10 constraints in to-

Table 3.2: Performance Comparison for Folded Cascode OTA

Algorithm	DE	BO-wEI	GASPAD	**DNN-Opt**
success rate	10/10	2/10	4/10	**10/10**
# of simulations	3200	>500	>500	**132**
Min power (mW)	0.75	0.91	0.72	**0.62**
Max power (mW)	1.53	1.62	1.75	**0.77**
Mean power (mW)	1.14	1.25	0.96	**0.71**
Modeling time (h)	NA	30	6.5	**0.6**
Simulation time (h)	54	2.7	2.7	2.7
Total runtime (h)	54	32.7	8.2	**3.3**

Table 3.3: SA-Latch Comparator Search Space

Parameter Name	Unit	LB	UB
L1-L2-L3-L4-L5-L6	μm	0.18	10
W1-W2-W3-W4-W5-W6	μm	0.22	50
CL_finger	integer	10	300

tal:

$$
\begin{aligned}
&\text{minimize Power} \\
&\text{s.t.} \quad \text{Set Delay} < 10 \ \text{ns} \\
&\qquad \text{Reset Delay} < 6.5 \ \text{ns} \\
&\qquad \text{Area} < 26 \ \mu\text{m}^2 \\
&\qquad \text{Input-referred Noise} < 50 \ \mu\text{Vrms} \\
&\qquad \text{Differential Reset Voltage} < 1 \ \mu\text{V} \\
&\qquad \text{Differential Set Voltage} > 1.195 \ \text{V} \\
&\qquad \text{Positive-Integration Node Reset Voltage} < 60 \ \mu\text{V} \\
&\qquad \text{Negative-Integration Node Reset Voltage} < 60 \ \mu\text{V} \\
&\qquad \text{Positive-Output Node Reset Voltage} < 0.35 \ \mu\text{V} \\
&\qquad \text{Negative-Output Node Reset Voltage} < 0.35 \ \mu\text{V}.
\end{aligned}
\tag{3.9}
$$

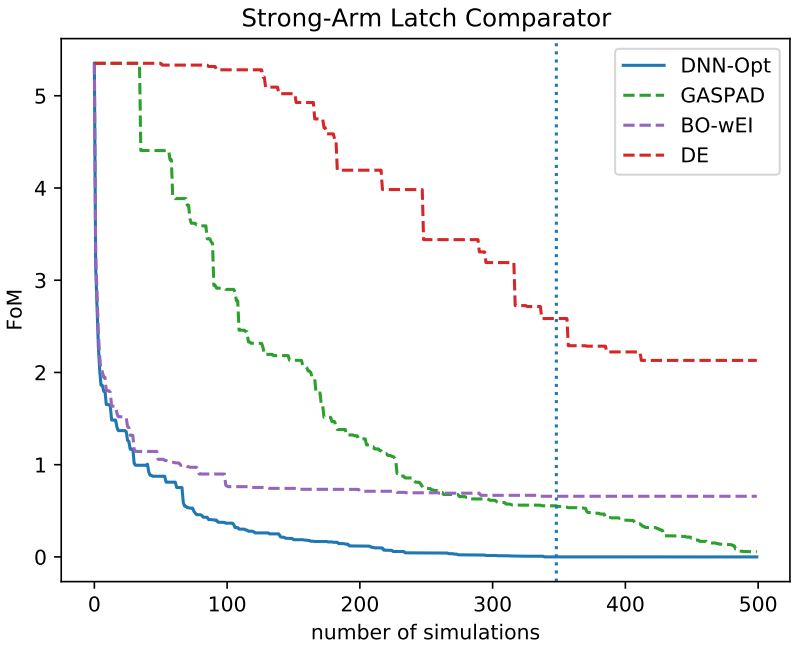


Figure 3.3: SA Latch Comparator FoM Convergence

The statistical results for all the reference algorithms are shown in Table-3.4. Due to relatively tighter constraints for SA-Latch Comparator, methods typically needed a larger number of simulations to converge. DDN-Opt is the only method that finds a feasible solution in all trials, and our method shows more than 30x efficiency compared to DE. GASPAD shows relatively competitive results, but DNN-Opt finds a solution with 25% better power consumption than successful runs of GASPAD. The runtime observations are similar to the folded cascode case.

FoM curves are shown in Figure 3.3 for different methods. DNN-Opt finds a feasible solution within 348 simulations, which is much earlier than the others. BO-wEI shows a similar convergence trend for initial iterations and then fails to model one of the constraints properly. Our observations showed that all the runs with the BO-wEI method were unable to meet input-referred noise, and some failed for set delay.

Table 3.4: Performance Comparison for SA Latch Comparator

Algorithm	DE	BO-wEI	GASPAD	DNN-Opt
success rate	5/10	0/10	6/10	**10/10**
# of simulations	>10000	>500	>500	**330**
min (μW)	2.98	NA	3.05	**2.50**
max (μW)	4.22	NA	3.75	**2.75**
mean (μW)	3.57	NA	3.45	**2.65**
Modeling time (h)	NA	17	3	**0.3**
Simulation time (h)	72	3.6	3.6	3.6
Total runtime (h)	72	20.6	6.6	**3.9**

3.4.2 Experiments on Industrial Circuits

We tested DNN-Opt on four industrial circuits designed at a very advanced technology node. These circuits were already in the process of manual sizing by expert analog designers and needed some fine-tuning. For these industrial circuits, we did not have access to other algorithms (DE, GASPAD, BO-wEI), and hence our baseline is with a commercial black-box optimizer based on Simulated Annealing. As will be demonstrated in this section, DNN-Opt performs well on large circuits and is not limited to small examples. Analog designers assisted in selecting permissible parameter ranges of the devices, considering layout impacts and process rules. For industrial cases, we identify critical devices based on Equation 3.6 for the failing constraints (f_i's of Equation 2.1). Note, MLParest (73) was used in the loop of DNN-Opt which helps analog designer estimate post-layout effects early in the design.

Inverter Chain: The first case is a simple inverter chain used mainly for tool development and flow testing. We used all the devices (8) in the four-stage inverter chain. There were only two specs, delay, and power.

Level Shifter: Sensitivity analysis identified ten critical devices impacting failing performances, and that led to a design space of 3.9×10^{15}. There were 60 total specs like delay, rise, fall, power, current, etc.

Low-Dropout (LDO) Regulator: We used sensitivity analysis to

Table 3.5: Performance Comparison on Industrial Circuits

Circuit	MOS	Nodes	Simulated Annealing (SA)	**DNN-Opt**
Inverter Chain	8	7	>1000	**90**
Level Shifter	1.2k	3.9k	1200	**195**
LDO	167k	2.8k	552	**112**
CTLE	173k	63k	587	**150**

Number of SPICE simulations shown in column SA and DNN-Opt for meeting constraints (lower is better).

identify six critical devices leading to search space of 1.6×10^{13}. The circuit had PSRR, Gain Margin, Phase Margin, DC Gain, GBW, etc., as part of nine constraints. The number of devices is high due to arrayed instances used by the analog engineer.

Continuous-Time Linear Equalizer (CTLE): Sensitivity analysis identified eight critical devices impacting failing performances. With design parameters and ranges identified by analog designers, we had a design space of 3.3×10^{25}. There were a total of 14 constraints like DC Gain, offset, Nyquist Gain, Fpeak, Peaking Max, Power, etc.

As illustrated in Table 3.5, DNN-Opt outperforms commercial optimizer available in the industry in terms of the number of simulations required to meet the constraints by 5x. We would like to emphasize that we can deal with fairly complex CTLE circuit by using 4x smaller number of costly SPICE simulations. Additionally, the optimal solution proposed by DNN-Opt consumed 8% lesser power than simulated annealing. Our examples represent real use cases where designers already spend several days worth of human time in fixing constraints. Had we started with designs without any knowledge of human designers baked-in, we would have seen even greater returns in sample efficiency like Section 3.4.1.

3.5 Summary

In this chapter, we presented DNN-Opt, a novel sample-efficient black-box optimization algorithm that combined the strengths of deep neural networks and reinforcement learning paradigm. We also give a recipe to extend our work for large circuits with thousands of devices. Our algorithm's effectiveness has been successfully demonstrated on various circuit building blocks and large industrial circuits, leading to 5–30x sample efficiency, while being able to find feasible solutions for all circuit sizing tasks and showing superior converge curves compared to other methods.

Chapter 4: APOSTLE: Asynchronously Parallel Optimization for Sizing Analog Transistors using DNN Learning[1]

In this chapter, we focus on analog problems with varying simulation costs. We are motivated by the fact that the initial evaluation of cheaper simulations can be used to make intermediate evaluations regarding the quality of the design. Then, based on this evaluation, the automation loop may avoid running the remaining costly simulations. We propose a Gaussian Process (GP) model to predict design ranks and study an efficiency theory to determine under what conditions the costly simulations are avoided.

4.1 Introduction

While learning-based methods (RL, DNN, Bayesian) target sample efficiency, most of these methods are sequential in nature and fall far from fully utilizing computational resources. In practice, SPICE simulations may be very costly, and the whole synthesis process is time budgeted, so a time-efficient

[1]This chapter is based on the following publication: Ahmet Faruk Budak, David Smart, Brian Swahn, David Z Pan, "APOSTLE: Asynchronously Parallel Optimization for Sizing Analog Transistors using DNN Learning," in ACM/IEEE Asia and South Pacific Design Automation Conference (ASPDAC), 2023. I am the main contributor in charge of problem formulation, algorithm development and experimental validations.

method is sought. One natural instinct is to insert a batch sampling technique to introduce parallelization and gain time efficiency, but the prior efforts are only limited by parallelization of Bayesian methods (55; 92; 32) which still suffer from computationally expensive GPR. Another way to gain efficiency is to make use of cheap simulations. In (87), a Multi-fidelity surrogate optimization technique is proposed where it uses fast RC simulations to reduce the cost of electromagnetic simulations. However, this method is not generalizable since not all expensive simulations have their cheap counterparts. Another approach is to impose custom conditions or procedures before running certain simulations. In (90), an example of this approach is applied as a PVT exploration strategy, which is useful for checking pass/fail conditions of specifications across several corners.

In this chapter, we introduce APOSTLE, an asynchronously parallelized efficient deep learning optimization scheme. We define a novel deep neural network (DNN) architecture and utilize it to be the engine of an efficient batch optimization procedure. The resulting algorithm can find similar quality designs in a much smaller runtime. The key features of the APOSTLE framework are below.

- A batch optimization framework for utilizing available computer resources.

- We introduce a deep learning sampling scheme that can be trained on a set of samples when some of them are partially evaluated.

- Circuit performance metrics are partitioned into those that can be computed by cheap simulations and those that require expensive simulations. Expensive simulations are performed for only those samples for which a probabilistic model predicts that they efficiently contribute to the optimization goal.

4.2 APOSTLE

In the context of analog sizing automation, the ultimate goal is to find one design that meets the spec list. The role of other designs is to provide guidance (modeling/sampling) in the search of that one solution. We are further aware that the performance of a design is evaluated by various simulation types (e.g., ac, dc, transient) that have various time costs. APOSTLE is motivated by the idea that one is not obligated to run all types of simulations for all samples, especially those with no benefit in the path to optimal.

APOSTLE categorizes various simulation types of the sizing problem into two: cheap simulations and expensive simulations. Every new sample is first evaluated based on its cheap simulations, and whether to run expensive simulations is decided based on the results of cheap simulations. The final

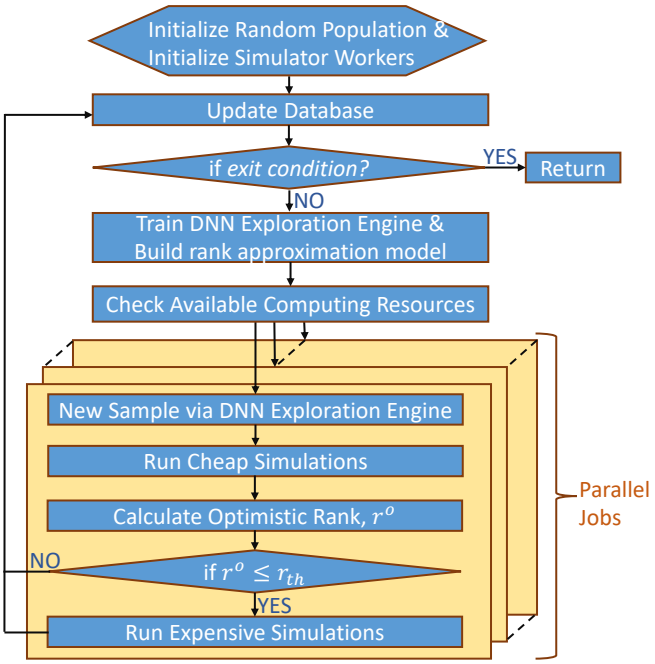


Figure 4.1: APOSTLE Framework

decision is to be made by comparing the partially evaluated sample's approximate rank among all designs vs. a theoretically calculated efficiency threshold rank value. APOSTLE is intended to be generalizable and does not require any correlation or hierarchy relation between cheap and expensive simulations.

4.2.1 Overall Batch Optimization Framework

The overall framework for APOSTLE is provided in Figure 4.1. Prior to core algorithm steps, we randomly sample N_{init} points from the design space to

form initial population and performance values. Then, the elite population and elite performance matrix can be constructed based on the FoM sorting. We will use the term *rank* throughout the chapter; it refers to the position of a design sample in a list sorted by FoM, with low rank corresponding to better FoM. After initialization, the flow continues with running B_{max} simulator workers in parallel.

At each iteration, t, a *DNN exploration engine* is trained. It has a custom training scheme to comply with the nature of our dataset and is used to propose sample points to query the optimization. Further, a *rank approximation model* is built to approximate rank of new designs after obtaining their performance evaluations from cheap simulations.

These steps are followed by creating parallel jobs based on the number of available computing resources, $B_t \leq B_{max}$. Each job is created independently and the availability of resources are checked per iteration so the parallelization scheme is asynchronous.

For each independent job, a new sample is proposed by the DNN exploration engine. The new sample is first evaluated on its cheap simulations, then an intermediate step is applied to decide whether to perform expensive simulations for the sample. This decision is made by comparing the *optimistic rank* of the sample, r^o, and a theoretically calculated efficiency threshold rank value, r_{th}. If a partially evaluated design's optimistic rank is better than the

theoretic efficiency threshold, that design is decided to be worth running expensive simulations. If not, expensive evaluations are not performed for the design. The details of the regression model for rank approximation is given in Section 4.2.3, and the theory behind calculating r_{th} is given in Section 4.2.4.

The final steps are to collect completed simulation results and update the population, i.e., the dataset. The loop will continue until the return condition is met. Typical return conditions are consuming the simulation budget or meeting a desired design quality (e.g., specs are met and no objective is defined).

4.2.2 DNN Exploration Engine

We utilize two networks as proxies for simulations. The *cheap critic network* is trained on the designs whose cheap simulations are completed. The *expensive critic network* is trained using expensive simulation results of a smaller sample set. This scheme allows us to utilize the simulation results at maximum capacity since there is no requirement for samples to be fully (and expensively) evaluated in order to be included in the training. Once the critics are trained and fixed, *actor network* is used to obtain the next sample.

In sizing literature, many actor-critic algorithms are used. The one closest to our approach is DNN-Opt (6). DNN-Opt uses a single critic network to approximate simulation values; however, APOSTLE employs individual

critic networks for the cheap and expensive type of performance metrics.

The input to critic networks is $(\mathbf{x}, \Delta\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^d$ is a vector containing the values of d design variables of the sizing problem. The output of the cheap critic is in $\mathbb{R}^{m_c}$, and the expensive critic is in $\mathbb{R}^{m_e}$ where m_c and m_e are the number of design specification values evaluated by each type of simulation category. In order to individually train critic networks, we employ a pseudo sample generation technique as described in the DNN-Opt method. To train the actor-network, we use the concatenation of values coming from both critics and also make use of the elite population to determine the next sample. We avoid repetition of the remaining details on training as there exists an elaborate work (6) dedicated to them. The figure that describes the sample exploration via modified actor-critic networks is given in Figure 4.2.

4.2.3 Finding Optimistic Rank

We quantify the quality of samples by their respective ranking to each other. The motivation behind our algorithm is to avoid running expensive simulations when even an optimistic guess for the design does not satisfy certain quality/rank requirements. Therefore we need a statistical model to obtain a rank approximation for a design after evaluating it on cheap simulations. Considering the fact that rank is a non-linear function of the cheap design performances, we utilize Gaussian Processes to build a model that predicts

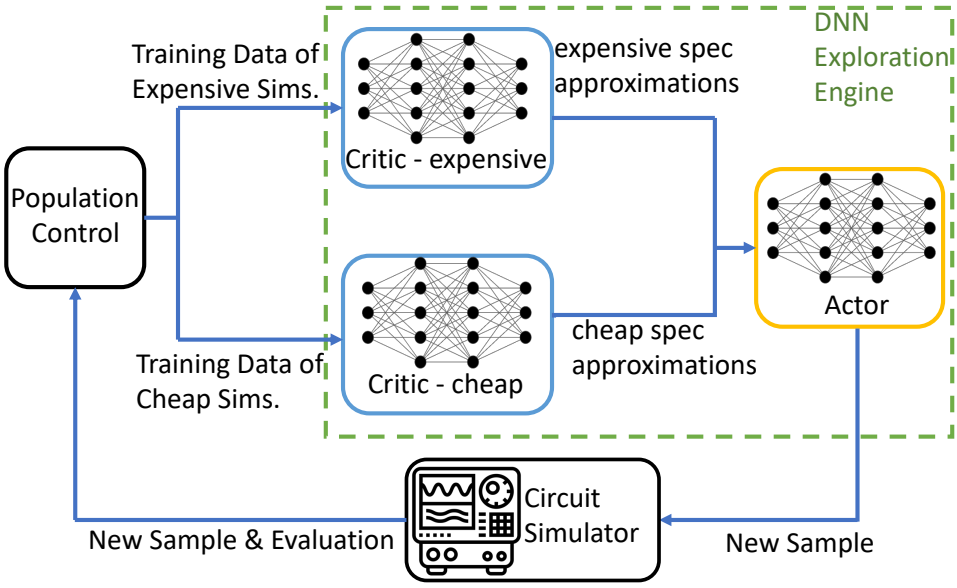


Figure 4.2: DNN Exploration Engine

the rank of a design.

4.2.3.1 Background on Gaussian Process

Gaussian Process (GP) is a kernel-based regression method that is capable of making mean and variance estimations to the untested regions of the design space based on existing samples (65). Given a pair of sampling points $\boldsymbol{x}$ and y, where $\boldsymbol{x}$ is a d-dimensional input and y is the corresponding scalar output, a standard GP model assumes that y is a noisy observation of the actual output $f(\boldsymbol{x})$, i.e., $y = f(\boldsymbol{x}) + \epsilon_y$. ,where $\epsilon_y \sim \mathcal{N}\left(0, \sigma_y^2\right)$ and σ_y is an independent identical noise variance. The probability of an observation y is

therefore, $P(y|f) = \mathcal{N}\left(f, \sigma_y^2\right)$.

To derive the posterior mean and variance, a Gaussian prior is chosen to be, $P(\boldsymbol{f}|X) = \mathcal{N}(\mu, K(X,X))$, where μ is the mean vector, X is an $n \times d$ matrix of input observations and $K(X,X)$ is the $n \times n$ covariance matrix, where n is the number of training data points.

With proper derivation (65), the gaussian mean and variance for a single test point $\boldsymbol{x}_*$ could be expressed as follows:

$$\begin{cases} \mu\left(\boldsymbol{x}_*\right) = \mu + \boldsymbol{k}\left(\boldsymbol{x}_*, X\right)\left[K(X,X) + \sigma_y^2 I\right]\left(\boldsymbol{y} - \mu\right) \\ \sigma^2\left(\boldsymbol{x}_*\right) = k\left(\boldsymbol{x}_*, \boldsymbol{x}_*\right) - k\left(\boldsymbol{x}_*, X\right) K^{-1} k\left(X, \boldsymbol{x}_*\right) \end{cases} \tag{4.1}$$

where $\boldsymbol{k}\left(\boldsymbol{x}_*, X\right)$ denotes the covariance vector between the test point $\boldsymbol{x}_*$ and all training data points X.

4.2.3.2 Optimistic Rank of A Partially Evaluated Design

We use fully simulated samples to build a GP model that predicts the rank of a sample based only on cheap simulation results. This model is used subsequently to decide whether to run expensive simulations for a new sample after running its cheap simulation. In summary, we have the followings:

$$\begin{aligned} \mathbf{r} &\leftarrow f\left(Y_{\mathrm{f}}^{\mathrm{tot}}\right) \\ \mathbf{r} &\sim GP\left(0, K(Y_{\mathrm{f,c}}^{\mathrm{tot}}, Y_{\mathrm{f,c}}^{\mathrm{tot}})\right) \end{aligned} \tag{4.2}$$

where f function includes the FoM calculations and sorting operations, and K is the covariance matrix based on the selected kernel function. $\mathbf{r}$ is

vector of ranks based on real evaluations and $Y_{\text{f,c}}^{\text{tot}}$ is cheap evaluations of samples that are fully evaluated.

Having a GP model built on cheap evaluations enables us to make inferences for design points that are partially evaluated. Using such inference, the optimistic rank of a cheaply evaluated design whose cheap specs are $y_{i,c}$ is:

$$\mathbf{r}_i^{\text{o}} = \mu_i(y_{i,c}) - \alpha * \sigma(y_{i,c}) \tag{4.3}$$

where μ_i and σ_i are obtained from the GP model and α is a hyperparameter typically selected from $\alpha \in [0, 2]$. This method is called lower/upper confidence bound method and is popularly used for optimization problems or bandit problems.

4.2.4 Theory: Efficiency Rank Threshold

We stated in the problem formulation that a design's specifications come from two categories of simulations: cheap and expensive. We defined (Section 4.2.3) a model to approximate the promising rank of designs after obtaining its cheap evaluations. Now, we study when it is better to invest for expensive simulations and fully evaluate a sample instead of exploring a new sample.

Optimization Impact: The goal of any optimization algorithm is to explore samples with good quality, i.e., samples with good ranks. Following

this intuition, we define a new metric called Optimization Impact (OI). The OI of a partially evaluated sample is zero, and the OI of a fully evaluated sample is calculated based on its rank, r:

$$OI(r) = (\max(0,\ N_{es} - r))^{\beta} \tag{4.4}$$

where N_{es} is the elite population size. OI is zero if the fully evaluated sample cannot rank inside the elite population, and it gets larger as the rank goes closer to 1. Hyperparameter $\beta \in [1, 2]$ is included to make the impact scale larger than linear to further reward the samples with better ranking. Throughout the optimization, the framework wants to maximize optimization impact per time.

We define R as the random variable denoting the rank of any new sample explored during the optimization such that $P(R = r)$ is the probability of getting a design with rank r. Further, let $F_{r'}$ be the cumulative distribution function of r being less than some value, r' where $F_{r\prime} = \sum_{r=1}^{r'} P(R = r)$.

We declared that our strategy, π, is first to run a cheap simulation then an approximate rank becomes available. If it is less than r_{th}, we will invest the time for expensive simulation to collect the optimization impact. The expected OI per time of this strategy for an arbitrary threshold value $r^{\star}$ is:

$$\frac{OI(\pi_{r^{\star}})}{t} = \frac{OI(r \mid r \leq r^{\star})}{T^{\star}} = \frac{\frac{\sum_{r=1}^{r^{\star}} P(R=r)(N_{es}-r)^{\beta}}{F_{r^{\star}}}}{T_c/F_{r^{\star}} + T_e} \tag{4.5}$$

where T_c and T_e are the time costs to complete cheap and expensive simulations, respectively. In the final expression, $T_c/F_{r^\star}$ is the expected time before getting a sample with a better ranking than $r^\star$ and $T_c/F_{r^\star} + T_e$ is the expected time before fully evaluating a sample with rank $r \leq r^\star$.

Once a sample is evaluated on cheap simulations, the remaining time to collect any impact is T_e, and its implied (if invested in expensive simulations) optimization impact per time is given as the following:

$$\frac{OI(r)}{T_e} = \frac{(N_{es} - r)^\beta}{T_e} \tag{4.6}$$

The decision to run expensive simulations after the cheap ones is made by comparing the expressions in Equation 4.5 and Equation 4.6. If the cheaply evaluated sample has a better expected implied impact than the average case, the framework decides to run expensive evaluations. However, if the cheaply evaluated sample's optimization impact per time is less than the expected value, the framework decides not to run it. Since we defined that the rank value where two expressions become equal to each other is r_{th}, we can calculate r_{th} as the r value solving the following expression:

$$r_{th} = \min_{r^\star} \left\{ r^\star \in \left(\frac{OI(r=r^\star)}{T_e} - \frac{OI(r|r \leq r^\star)}{T^\star} \geq 0 \right) \right\} \tag{4.7}$$

Considering that r_{th} can only take integer values defined in a finite set, the practical value of r_{th} can be determined by a simple grid search.

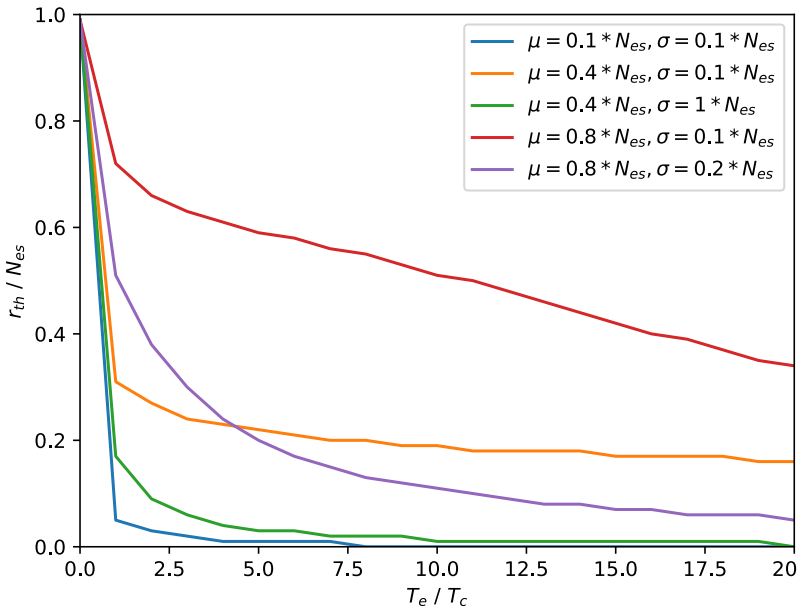


Figure 4.3: Rank Threshold Values of Selected Cases

The remaining element of our method is to approximate a model for DNN Engine sample rank distribution. Since we have a built GP model for rank approximations, the history of mean and variance approximations of every new sample (after cheap evaluation) is used to approximate the statistics of the sampling engine with mean μ and standard variation σ.

We studied r_{th} values in various environments to further demonstrate. For various new sample distributions and various values of expensive and cheap simulation ratios, the collected results are given in Figure 4.3. It can be observed from the graph that resulting r_{th} values satisfies several desired features:

1) As T_e goes to 0, r_{th} goes to N_{es}. If the expensive simulations come for free, anything falling into the elite population is simulated.

2) As T_e goes larger, r_{th} gets smaller. When the relative cost of expensive simulations is too large, only exceptionally-looking designs are simulated.

3) As the model gets better (smaller μ), r_{th} goes smaller. If we have a sampling model that explores good-quality designs, the r_{th} adapts to raise expectations before running expensive simulations.

4.3 Experiments

To demonstrate the efficiency of APOSTLE, we test it on three examples, two of which are smaller building blocks, and one is a larger circuit from a design house. The performance values from expensive simulations are marked with (*), and others are cheap to evaluate.

We compare our algorithm with three other methods: a) DNN-Opt (6), an RL-inspired DNN algorithm b) Bayesian Optimization with weighted Expected Improvement (BO)(57), which is a modified version of Bayesian Optimization for constrained problems, and c) GASPAD method(41), a surrogate model (GP) assisted evolutionary framework.

We repeated small experiments five times and averaged results to account for the randomization involved in the methods. All experiments are

initialized with the same seed and with a randomly sampled initial population of size $\min(5 \times d, 150)$ where d is the number of design variables. APOSTLE is configured with $B_{max} = 8$, $\alpha = 1.5$ and $\beta = 1.4$.

4.3.1 Testcases and Optimization Curves

To discuss each experiment, we will provide constraint-based FoM evaluations (i.e., $w_0 = 0$), where we compare all algorithms in terms of the best FoM found with respect to total optimization time (including modeling and simulation time).

Folded Cascode OTA: The first test case is a two-stage folded-cascode Operational Transconductance Amplifier (FC OTA). It has 20 design variables consisting of transistor lengths and widths, and capacitor values. 19 transistors are constrained by the saturation margin requirements, and the remaining spec requirements are given as follows:

$$\begin{array}{ll} \text{minimize Power} & \\ \text{s.t.} \quad \text{DC Gain} > 60 \ \text{dB} & \text{Settling Time}^\star < 30 \ \text{ns} \\ \quad \text{CMRR} > 80 \ \text{dB} & \text{Saturation Margin} > 50 \ \text{mV} \\ \quad \text{PSRR} > 80 \ \text{dB} & \text{Unity Gain Freq.}^\star > 30 \ \text{MHz} \\ \quad \text{Out. Swing} > 2.4 \ \text{V} & \text{Out. Noise}^\star < 30 \ \text{mV}_{\text{rms}} \\ \quad \text{Static error}^\star < 0.1 & \text{Phase Margin}^\star > 60 \ \text{deg.} \end{array} \tag{4.8}$$

Figure 4.4 includes the FoM curve of APOSTLE and other baseline methods with respect to time.

Strong-Arm Latch Comparator: The second test is a strong-arm latch (SA) comparator with 13 design variables. This example has 3 simulation

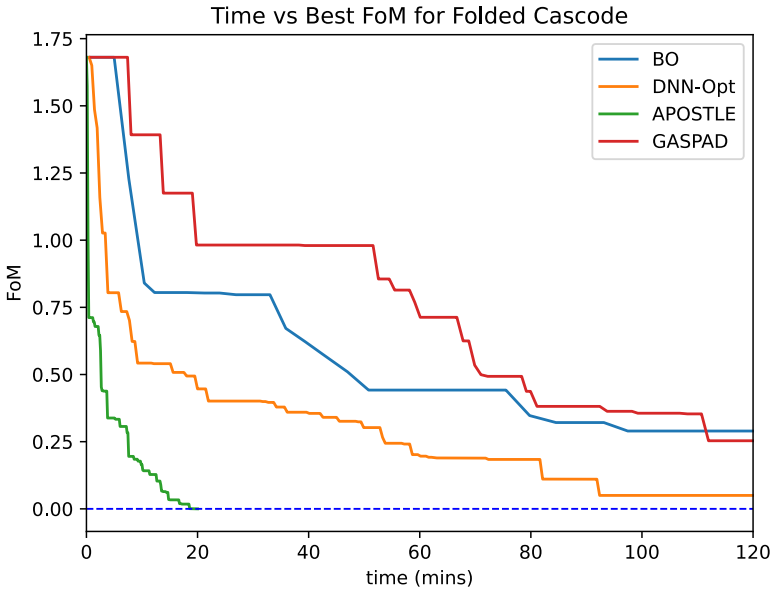


Figure 4.4: Folded Cascode OTA FoM Convergence w.r.t. time

types: transient, noise, and offset. Offset simulation dominates others in time since it is done by a Monte Carlo simulation. The constrained optimization problem consists of 10 constraints in total:

$$
\begin{aligned}
&\text{minimize } \text{Power}^\star \\
&\text{s.t.} \quad \text{Set Delay}^\star < 10 \ \text{ns} \\
&\qquad \text{Reset Delay}^\star < 6.5 \ \text{ns} \\
&\qquad \text{Input-referred Noise} < 50 \ \mu\text{Vrms} \\
&\qquad \text{Differential Reset Voltage}^\star < 1 \ \mu\text{V} \\
&\qquad \text{Differential Set Voltage}^\star > 1.195 \ \text{V} \\
&\qquad \text{Pos.-Integration Node Reset Voltage}^\star < 60 \ \mu\text{V} \\
&\qquad \text{Neg.-Integration Node Reset Voltage}^\star < 60 \ \mu\text{V} \\
&\qquad \text{Pos.-Output Node Reset Voltage}^\star < 0.35 \ \mu\text{V} \\
&\qquad \text{Neg.-Output Node Reset Voltage}^\star < 0.35 \ \mu\text{V}.
\end{aligned}
\tag{4.9}
$$

Figure 4.5 includes the FoM curve of APOSTLE and other baseline methods with respect to time.

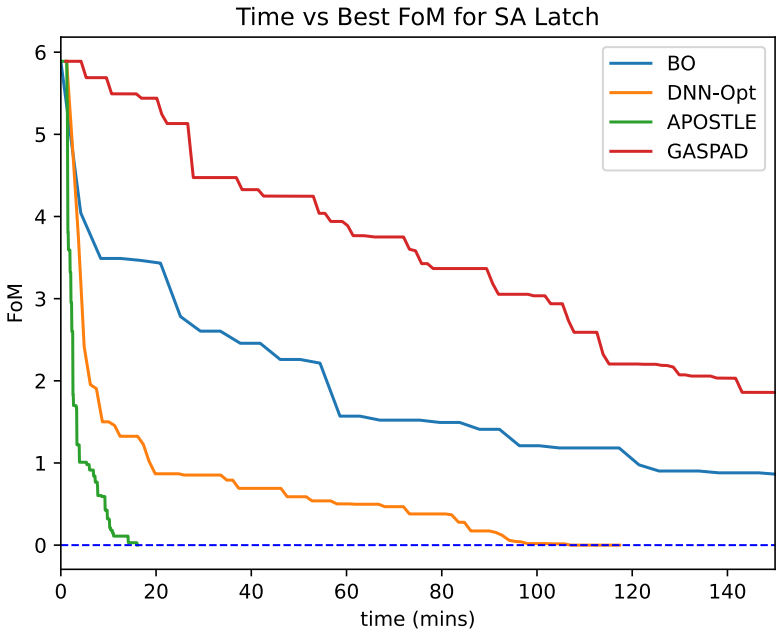


Figure 4.5: SA Latch Comparator FoM Convergence w.r.t. time

Programmable Gain Amplifier: The final test case is an advanced node Programmable Gain Amplifier (PGA). This is the largest test case and has 58 independent design variables. The optimization problem has no objective and is defined as follows:

$$\begin{array}{lll} \text{minimize None} & & \\ \text{s.t.} & \text{Gain}^{\star} > 0.99 & \text{Bandwidth} > 2.1 \ \text{Mhz} \\ & \text{THD}^{\star} < -95 & \text{Phase Margin} > 64 \ \text{deg} \\ & \text{Offset} < 24.5 \ \mu\text{V} & \text{Total Noise} < 6.8 \ \mu\text{Vrms} \\ & \text{A} > 90 \ \text{dB} & \end{array} \tag{4.10}$$

Figure 4.6 includes the FoM curve of APOSTLE and other baseline methods with respect to time.

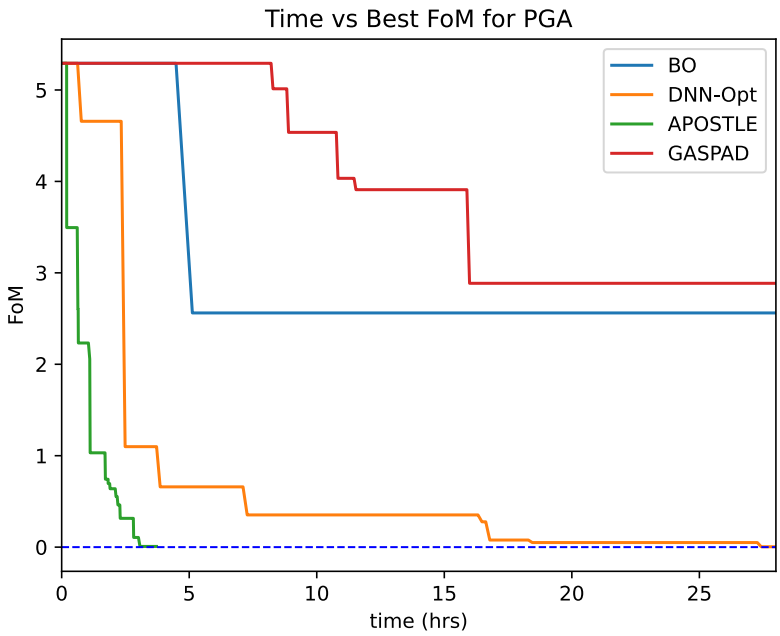


Figure 4.6: PGA FoM Convergence w.r.t. time

4.3.2 Combined Results and Discussion

To better interpret the results, we compile some important metrics in Table 4.1. We further focus on comparing APOSTLE and DNN-Opt as they are competitive algorithms compared to the others.

In general, parallelization results in some degradation in learning since the parallelized scheme has to take multiple steps using the same amount of information. To analyze this effect, we report the average number of samples used to find feasible solutions. The percentage overhead of visited designs by APOSTLE compared to DNN-Opt is between $[-25\%, +16\%]$. Based on this data, we do not observe a significant overhead issue. In fact, there is even a case (OTA) where APOSTLE finishes the same task with a smaller total number of actions.

We also study the real-time efficiency of APOSTLE. Total time data

in the table suggests that APOSTLE's time efficiency varies between [6.7x to 9x], which is a reasonable figure given $B_{max} = 8$.

The following lines in Table 4.1 show the mean objective value reached by the algorithms when they are given the same time and APOSTLE overperformed DNN-Opt in both cases.

The last five rows in Table 4.1 are informative data regarding the use of the defined simulation strategy. This data is further helpful to distinguish how much the simulation strategy contributes to overall efficiency. We calculate the total CPU time units spent for each problem by normalizing the CPU time of cheap simulations to 1. The last two rows display the total CPU time used by DNN-Opt and APOSTLE, where APOSTLE provided [17% to 30%] reduction in total computing time.

Table 4.1: Important Metrics for Result Discussion

Testcase	FC OTA	SA Comp	PGA
# of samples APOSTLE	121	93	223
# of samples DNN-Opt	162	80	216
total time APOSTLE	18mins	14mins	3.7hrs
total time DNN-Opt	120mins	125mins	27hrs
objective val. APOSTLE	0.67 mW	2.6 μW	NA
objective val. DNN-Opt	1.3 mW	3.3 μW	NA
T_e/T_c	1.3	5	10
# of bypassed ES	11.8	32.3	54
% of bypassed ES	9.8%	31.6%	24.2%
Tot. CPU units APOSTLE	263	396	1913
Tot. CPU units DNN-Opt	372	480	2376

4.4 Summary

In this chapter, we presented a novel learning-based batch optimization algorithm, APOSTLE. We tailored concepts from learning and statistics to meet the needs of analog sizing. The time efficiency of the proposed algorithm is demonstrated on various circuit examples. Being configured with eight workers, APOSTLE provided 6.7x-9x real-time efficiency and 17% to 30% total computation time reduction compared to baseline DNN-Opt, proving the success of the proposed asynchronous parallelization scheme and the idea of simulation bypassing.

Chapter 5: Layout-Aware Analog/Mixed-Signal Design Automation[1]

5.1 Introduction

There have been efforts to enhance automatic sizing with parasitic or layout awareness. Some existing methods use post-layout performance as the optimization objective of simulation-based analog sizing algorithms so that the device sizing can take layout effect into consideration. Their methodologies mainly fall into two categories.

1. *Layout-aware sizing with procedural layout generation.* An automatic analog sizer is combined with a procedural layout generator. The flow generates the layouts based on a template for a set of sizing, and the post-layout performance is obtained from the generated results.

2. *Parasitic-aware sizing with parasitic prediction.* A machine learning

[1]A part of this chapter is based on the following publication: Ahmet Faruk Budak, Keren Zhu, Hao Chen, Souradip Poddar, Linran Zhao, Yaoyao Jia and David Z. Pan, "Joint Optimization of Sizing and Layout for AMS Designs: Challenges and Opportunities," in ACM International Symposium on Physical Design (ISPD), 2023. Remaining part of this chapter is based on the following accepted work: Ahmet Faruk Budak, Keren Zhu, David Z Pan, "Practical Layout-Aware Analog/Mixed-Signal Design Automation with Bayesian Neural Networks," accepted to ACM/IEEE International Conference on Computer-Aided Design, 2023. I am the main contributor in charge of problem formulation, algorithm development and experimental validations.

model to predict layout parasitics from pre-layout netlists is trained. The automatic analog sizing flow can then use the prediction model as a vehicle to model the layout effect.

The rest of this section provides an overview of the recent academic developments.

5.1.1 AMS Sizing With Procedural Layout Generation

A procedural analog layout generator uses a manually scripted layout generator to enable automatic sizing to generate the layouts for a template schematic. Both two recent methods, AutoCkt (70; 69) and BagNet (28), leverage the BAG (16) framework as the layout generator.

AutoCkt educates RL agents to be knowledgeable of the full design space. It uses BAG to enable layout awareness. Consequently, the agent is able to select the optimal course of action at a particular moment in the design process. Multiple trajectories teach the agent how to create a circuit. The agent either reaches the target or ends the trajectory because the maximum number of steps has been reached. Following training, the agent is assigned to an unseen target. The more successful the training, the higher the success rate agent can achieve during the deployment.

BagNet, on the other hand, is based on an evolutionary algorithm with

a deep neural network (DNN)-based oracle to boost the search efficiency. It contains three core components: (1) an evolutionary engine to generate offspring, (2) the BAG layout generator, and (3) a DNN model acting as an oracle. Based on the existing population, the evolutionary algorithm produces the following generation of children. The DNN-based oracle evaluates the produced candidates and eliminates those with lower scores. After the discriminating step, the resulting offspring are implemented in layouts using BAG and simulated to acquire the metrics. The simulated performance is then utilized to fine-tune the oracle and direct the next generation in the evolutionary process.

The procedural layout generator-assisted analog sizing can obtain accurate post-layout performance. However, they require manually scripted layout templates.

5.1.2 AMS Sizing With Parasitic Prediction

Recent advances in machine learning have enabled statistical and data-driven techniques to predict layout parasitics directly from circuit schematics with high accuracy. ParaGraph (67) converts circuits into heterogeneous graphs and uses a graph neural network (GNN) model to predict net parasitic capacitance. MLParest (74) uses Random Forest models to predict resistance and lumped parasitic capacitance.

The work (51) leverages the Paragraph prediction model to enable

parasitic-aware sizing. It further proposes an improved performance surrogate model using graph embeddings from the pre-trained parasitic graph neural network as additional parasitic information. The ParaGraph GNN model encodes the latent information of the circuit parasitics with graph embeddings. With the performance model, With the surrogate model, the automatic sizing framework can therefore enable parasitic awareness and mitigate the layout effect.

The ML-based parasitic prediction and performance prediction provides an efficient vehicle to take layout effect into the sizing loop. However, the accuracy of predicting layout effects is still a challenge without layout implementation in both the parasitic prediction (96) and performance prediction (52) tasks.

5.2 Preliminaries

5.2.1 Analog Layout Automation

The physical layout implementation stage in the current analog IC design is still mostly manual, labor-consuming, and error-prone, placing constraints on turnaround time. Procedural layout generation and optimization-based layout synthesis are two paradigms that describe AMS Layout automation techniques (13).

Utilizing pre-designed parameterized layout templates, procedural lay-

out generators migrate layouts for various manufacturing technologies and device sizes. Berkeley Analog Generator (BAG) (16) is a framework of procedural layout generation. This method typically requires designers to parameterize device placement and route (such as routing topology, metal layer, and via cuts), allowing designers to modify device sizing. Nonetheless, it needs a large amount of human labor to develop generic layout templates or PCELLs, where device layouts are still manually coded to be placed and routed.

In generating layouts, optimization-based techniques employ place-and-route (PNR) algorithms to optimize area, power, and certain performance metrics. These methods have varying degrees of generality and are designed for various application scenarios. In contrast to template-based layout creation, optimization-based methods produce fully automated layout solutions without the need for additional effort to build layout templates. To assure layout quality, it is necessary to recognize constraints such as device symmetry, building block symmetry, and common-centroid matching.

Developing open-source end-to-end frameworks, such as MAGICAL (89) and ALIGN (38), is an emerging trend in optimization-based AMS circuit layout automation. MAGICAL and ALIGN both include a complete module generator, analog placer, and detailed router. They have demonstrated success in automating building block-level analog circuits with minimal or no human intervention. These general-purpose analog layout generators can gen-

erate the layouts in automatic sizing flow and, therefore, analog sizing to be layout-aware. In this chapter, we leverage the open-source MAGICAL framework as a bridge to study the joint optimization of sizing and layout for AMS designs.

5.2.2 MAGICAL

MAGICAL is an open-source end-to-end framework for AMS layout synthesis that aims to build layouts from netlists with no-human-in-the-loop automatically. In a few years, the MAGICAL system has evolved from the initial version with limited capability (89) to the silicon-proven MAGICAL 1.0 release (12).

MAGICAL automatically extracts layout constraint, generates the device layouts, places the modules, and routes the wires. Figure 5.1 illustrates the flow of MAGICAL for generating the layout for a block-level analog circuit. The automatic symmetry constraint generation is based on heuristic algorithm for device-level constraints (89) and graph similarity (50) for the system-level. The placement engine is based on a non-linear programming-based global placer and a linear programming-based legalizer (95). The MAGICAL detailed router is based on an obstacle-aware path-finding algorithm (14). The paper (94) gives a detailed overview of the MAGICAL algorithm.

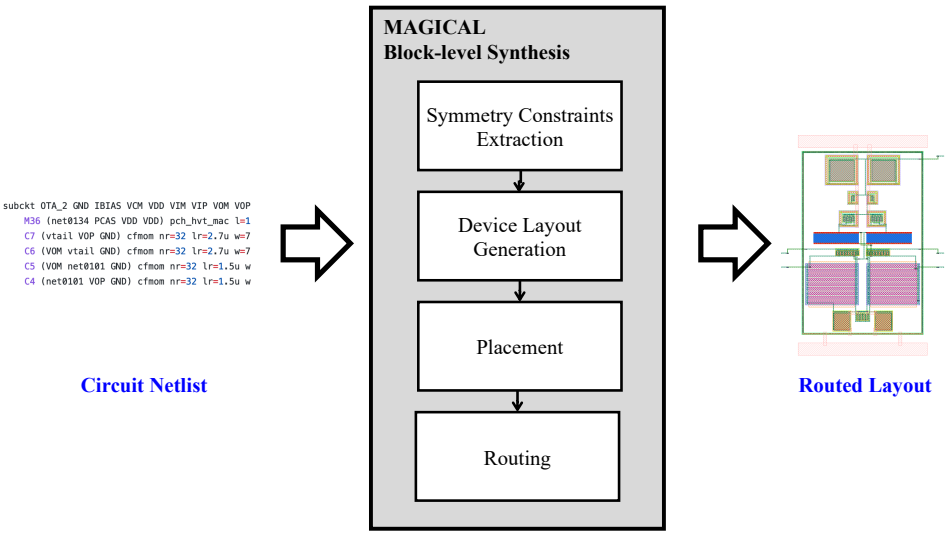


Figure 5.1: The MAGICAL flow for block-level layout generation.

5.3 Layout Integrated Sizing: A Case Study

The majority of the research to automate analog design flow (Fig 1.3) focuses on individual phases and does not address the effects of preceding and following phases on each other. As reviewed in the previous chapters, analog automation literature is populated with mostly sizing-only or layout-only efforts. This practice has three major undesired consequences:

1. Being agnostic of other phases damages the reliability of automation.

2. The total scope of optimization is restrained as certain phases are excluded from the automated flow.

3. Isolating these phases does not reflect the industry practice and, therefore, discourages the adaptation of such tools by the industry.

In this section, we quantitatively study the importance and potential of the layout-aware analog sizing framework. We combine the state-of-the-art analog sizing algorithm, DNN-Opt (6; 8), and the general-purpose analog layout generator, MAGICAL (12; 89; 94). Leveraging the joint framework, the analog sizer can obtain accurate post-layout circuit performance with the requirement of manually scripted layout templates. We conduct a case study on a representative analog circuit and evaluate the joint framework.

5.3.1 Experiment Design and Setup

In order to demonstrate the importance of layout effects on the final performance, we perform experiments on a Miller OTA circuit designed in 40nm technology (Fig. 5.2). The transistors of the circuit are parameterized so that an automation flow to optimize its performance can be conducted. The optimization problem has 17 independent design variables to size length, width, and number of fingers of the transistors.

The optimization problem for the Miller OTA consists of an objective and several performance constraints and is expressed as below:

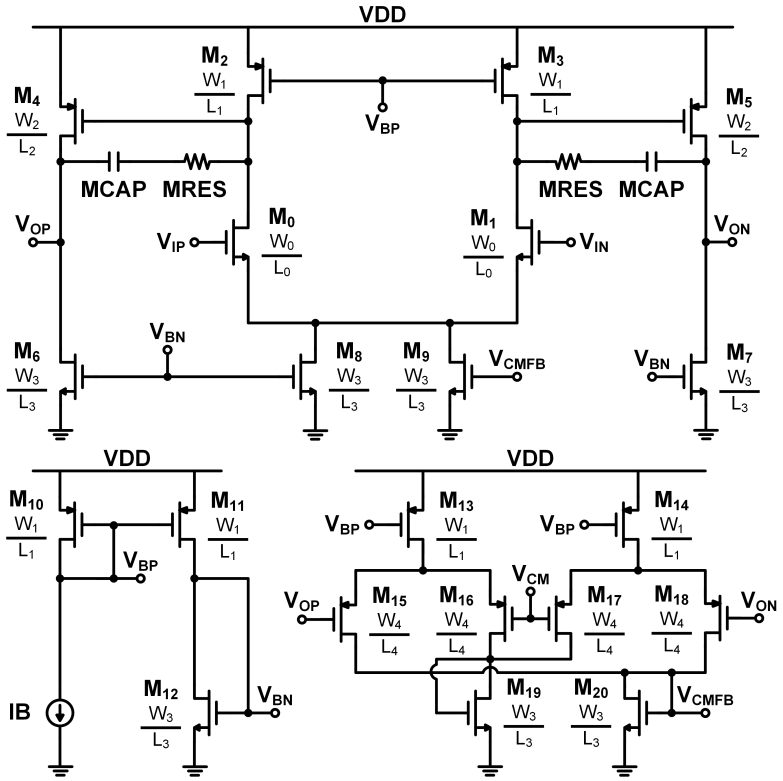


Figure 5.2: Schematic of the Miller OTA

$$
\begin{array}{ll}
\text{minimize Power} & \\
\text{s.t.} \quad \text{DC Gain} > 45 \ \text{dB} & \text{Settling Time} < 100 \ \text{ns} \\
\quad \text{CMRR} > 55 \ \text{dB} & \text{Saturation Margins} > 50 \ \text{mV} \\
\quad \text{PSRR} > 55 \ \text{dB} & \text{Unity Gain BW.} > 40 \ \text{MHz} \\
\quad \text{Out. Swing} > 1 \ \text{V} & \text{RMS Noise} < 400 \ \text{uV}_{\text{rms}} \\
\quad \text{Static error} < \%2 & \text{Phase Margin} > 60 \ \text{deg.}
\end{array}
\tag{5.1}
$$

Our experimental study includes the following two steps:

1. We first use the sizing automation tool DNN-Opt to optimize the performance metrics. This step is based-on schematic-level electrical simulations and is layout agnostic. We then auto-generate the layout of the optimized design using MAGICAL and run post-layout simulations to obtain performance values, including the layout effects.

2. We modify DNN-Opt algorithm to optimize the post-layout performance of the circuit. To facilitate this approach, we utilize MAGICAL as the layout generator.

5.3.2 Experiments with Layout Agnostic Loop

In this part of the study, we investigate how much performance degradation is introduced to the layout agnostic flow, i.e., the amount of suffering from completely delayed layout considerations during sizing. To do a fair analysis, we do not use the designer's after-layout intent to formulate the problem, but we refine it using the designer's schematic-level performance. This way, we match the designer's overdesign effort during sizing.

We compile the experiment results in Table 5.1. We included the new objective and constraint values used for the schematic-level optimization. DNN-Opt was able to provide a design that is very close to satisfying all design intent except for a minor miss for output swing. We further observe that the schematic-level result found by DNN-Opt overperforms designer-crafted

Table 5.1: DNN-Opt (Schematic) vs. Designer Performance Comparison Based on Pre-Layout Performance

Schematic Optimized	Intent	DNN-Opt	Designer
Power (mW)	minimize	**0.51**	0.53
Output Swing (V)	≥ 1	**0.99***	0.92
Gain (dB)	≥ 46	**48.1**	46.7
CMRR (dB)	≥ 55	**66.1**	56.2
PSRR (dB)	≥ 55	**63.7**	55.8
Phase Margin (deg)	≥ 57	**62.1**	57.2
RMS Noise (uV)	≤ 400	**380**	390
Rise Time (ns)	≤ 50	**21.3**	22.2
Static Error (%)	≤ 1.2	**1.08**	1.19
UGB (MHz)	≥ 85	**85.9**	85.0

design in every metric.

After obtaining optimized results on schematic-level simulations, we use MAGICAL to create layouts for both DNN-Opt and designer solutions. Layout generation is followed by parasitic extraction and post-layout simulations to obtain the post-layout performance of the Miller OTA test case. The post-layout simulation results of DNN-Opt optimized design and designer-created design are collected at Table 5.2.

We observe that all other performance metrics degrade in value except for the phase margin and rms noise. The metrics such as the Gain, CMRR, and PSRR are severely reduced for DNN-Opt generated design after layout effects.

Table 5.2: DNN-Opt (Schematic) vs. Designer Performance Comparison Based on Post-Layout Performance

Schematic Opt + Layout	Intent	DNN-Opt	Designer
Power (mW)	minimize	0.53	0.53
Output Swing (V)	≥ 1	0.96*	**0.97***
Gain (dB)	≥ 45	17.9*	**47.8**
CMRR (dB)	≥ 55	25.6*	**53.7***
PSRR (dB)	≥ 55	25.7*	**55.6**
Phase Margin (deg)	≥ 60	**75.1**	69.9
RMS Noise (uV)	≤ 400	370	370
Rise Time (ns)	≤ 100	**23.0**	110*
Static Error	≤ 2	**1.07**	2.52*
UGB (MHz)	≥ 40	41.6	**42.0**

Together with output swing, these metrics can not satisfy the design intent after layout. On the other hand, the designer's design demonstrates much better resilience against the layout effects. Its Gain and PSRR values meet the constraint threshold, and CMRR is very close to meeting the threshold. Overall, our analysis shows that the performance values of a circuit deviate significantly once a layout is generated. Therefore, the layout effects must be included in the parameter optimization phase in order to generate a robust sizing solution.

5.3.3 Layout in the Loop Automation Flow

To include the post-layout effects on the performance during sizing, we tailor the classical sizing flow. Instead of optimizing the design variables based on the schematic level simulations, we utilize the layout automation tool MAGICAL to modify performance evaluation steps. The suggested flow is shown in Figure 5.3. To obtain the post-layout performance of each new design, first, an automated layout is generated via MAGICAL. This step is followed by parasitic extraction, and circuit simulations are run on the updated netlist with parasitic elements.

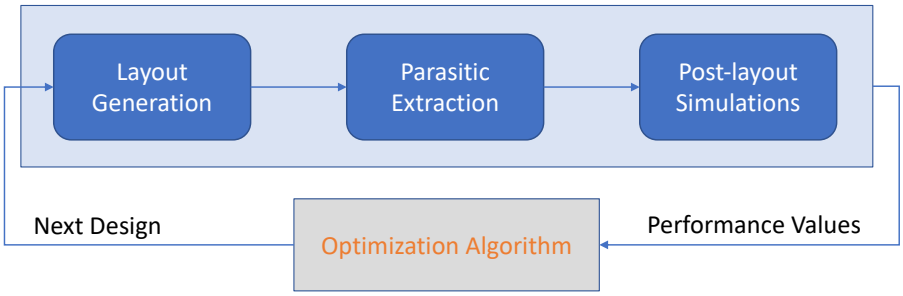


Figure 5.3: Post-Layout Performance Based Optimization

We evaluate the effectiveness of the layout-in-the-loop optimization flow by revisiting the Miller OTA case. The same optimization algorithm DNN-Opt is used to optimize the same set of parameters using the post-layout performance values instead of the schematic (pre-layout) simulations. The optimized post-layout performance values and their comparison to designer

performance are included in Table 5.3.

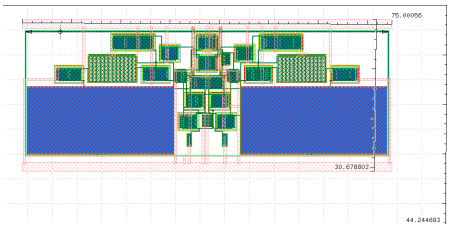
Table 5.3: DNN-Opt (Post-Layout) vs. Designer Comparison on Post-Layout Performance

Post-Layout Optimized	Intent	DNN-Opt	Designer
Power (mW)	minimize	**0.39**	0.53
Output Swing (V)	≥ 1	**1.11**	0.97*
Gain (dB)	≥ 45	46.1	**47.8**
CMRR (dB)	≥ 55	**56.7**	53.7*
PSRR (dB)	≥ 55	**58.9**	55.6
Phase Margin (deg)	≥ 60	**70.7**	69.9
RMS Noise (uV)	≤ 400	370	370
Rise Time (ns)	≤ 100	**26.9**	110*
Static Error	≤ 2	**1.2**	2.52*
UGB (MHz)	≥ 40	31.3*	**42.0**

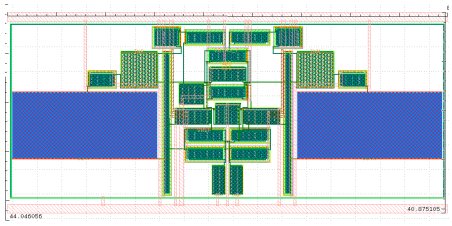
The tool-generated design satisfies all constraints except the unity gain bandwidth (UGB). Considering that the designer's design fails to meet four metrics proves that the proposed flow is very effective. Further, the layout-in-the-loop sized solution overperforms the designer's solution in seven metrics and falls behind only in two metrics (Gain, UGB).

We further elaborate our analysis by including the layouts generated by MAGICAL in Figure 5.4.

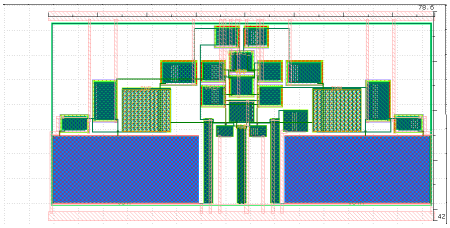
Analysis for Figure 5.4a: It is the most compact layout among all generated layouts. The designer has used their experience to provide a layout-



(a) Designer optimized design's layout



(b) Post-layout performance-based optimized design's layout



(c) Schematic-level performance-based optimized design's layout

Figure 5.4: Layouts for different design flows

friendly design. The total area is around $75\mu m \times 30\mu m$.

Analysis for Figure 5.4b: Despite the excellent performance, the layout has some issues in terms of compactness and area consumption compared to the designer's layout. Compared to the designer's layout, the layout-in-the-loop DNN-Opt generated result consumes around 60% larger area as its total area is $90\mu m \times 40\mu m$. Additionally, the lack of dummy devices for critical devices increases the risk of performance degradation in the actual silicon due to fabrication-related non-linearities and issues that are not modeled in the simulation. To improve the final design quality, the flow can be optimized

to decrease area consumption and automatically generate dummy devices to increase reliability.

Analysis for Figure 5.4c: The area consumed by the schematic-level performance-optimized design is around $79\mu m \times 42\mu m$ and 47% larger than designer layout. Also, it resulted in some key devices with abnormal sizes, such as long channel lengths and very short widths. This can cause deviation in these devices' transconductance and output resistance when post-layout effects are introduced into the simulation, leading to a significant degradation in performance parameters such as gain, CMRR, and PSRR. The abnormal-sized devices may also increase parasitic capacitance and resistance, further degrading the UGB (Unity Gain Bandwidth).

5.3.4 Improving Automated Layout Area

The case study reveals that optimizing performance without any layout consideration may result in a significant penalty for the generated layout area. To mitigate this problem, we modify the optimization setup and include the total layout area as a constraint metric. More specifically, an area constraint that matches the designer's area (generated by MAGICAL) is added to Equation 5.1.

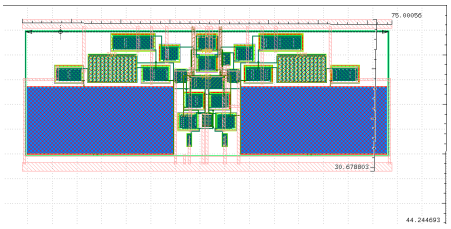
We share the updated performance comparison between designer and automated solutions with (w/) and without (w/o) the layout area considered

Table 5.4: DNN-Opt vs. Designer Post-Layout Performance Including Area

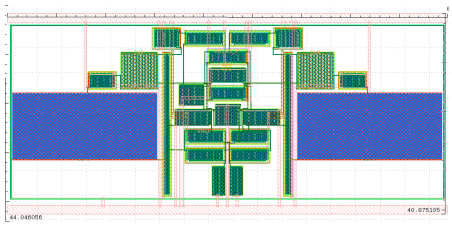
Post-Layout Optimized	Intent	DNN-Opt(w/o Area)	DNN-Opt(w/ Area)	Designer
Power (mW)	minimize	**0.39**	0.5	0.53
Output Swing (V)	≥ 1	1.1	**1.3**	0.97*
Gain (dB)	≥ 45	46.1	47.5	**47.8**
CMRR (dB)	≥ 55	56.7	**64.9**	53.7*
PSRR (dB)	≥ 55	**58.9**	57.0	55.6
Phase Margin (deg)	≥ 60	70.7	**78.4**	69.9
RMS Noise (uV)	≤ 400	**370**	380	**370**
Rise Time (ns)	≤ 100	26.9	**21.2**	110*
Static Error	≤ 2	1.2	**1.1**	2.52*
UGB (MHz)	≥ 40	31.3*	37.8*	**42.0**
Area (um^2)	≤ 2250	3600	2275	**2250**

(Table 5.4). It is observed that DNN-Opt is able to almost match with the designer's generated layout area. Compared to the previous DNN-Opt solution where the layout area is not considered, the area has improved by 37%. Further, no other performance metric is sacrificed for this improvement. In fact, several constraints, such as CMRR, phase margin, and UGB have further improved. We observed 20% better UGB and 12% increased phase margin. One drawback of the updated sizing result is the power consumption. The run with area constraint now requires more power, but it still has a better power figure compared to the designer's.

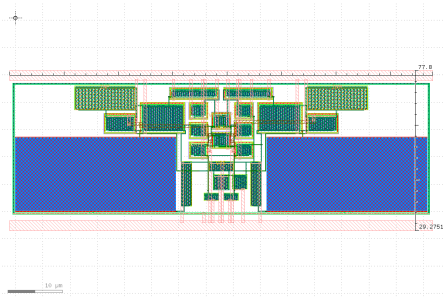
Figure 5.5 includes a comparison for considering layout area during post-layout-based optimization. The area-optimized DNN-Opt solution is shared in Figure 5.5c, demonstrating that the new approach mitigated most of



(a) Designer optimized design's layout



(b) Post-layout DNN-Opt without Area Consideration



(c) Post-layout DNN-Opt with Area Consideration

Figure 5.5: Effect of Including Layout Area During Optimization

the issues we observed during the previous run. The new layout does not have abnormally sized devices and is more compact than the run with no layout area consideration.

5.3.5 The Cost of Layout-Aware Sizing

Finally, we want to highlight the cost aspect of the adapted approaches. Our study revealed that - for our particular test case - the post-layout DNN-Opt is 9 times more computationally expensive than the schematic-level DNN-

Opt. Further, DNN-Opt spent 5 times higher time optimizing the final layout area and other metrics compared to the case with no layout consideration. This summarizes that optimizing the post-layout performance with area consideration is much more computationally heavy than traditional schematic performance-based sizing. Therefore, we need efficient methods to optimize the post-layout performance for improved practicality.

5.4 Practical Layout-Aware Sizing via Bayesian Neural Networks

Recent research trend in analog sizing introduces Machine Learning (9) to simulation-based methodology. However, the literature review reveals that most of these methods require thousands of simulation data to train Deep Neural Network (DNN) models that approximate the relations between the design variables and the performance metrics. Therefore, the practicality of these algorithms is severely reduced when the optimization task has a high simulation cost. For example, drawing/generating the layout, extracting the parasitics, and running post-layout simulations is typically an expensive procedure. Therefore, optimization algorithms designed for schematic-level sizing can not be adapted by simply changing how data is generated.

This section presents a Machine Learning-based simulation-in-the-loop automation method for the AMS design problem. Overall, we formalize two

stand-alone recipes for schematic-level sizing and post-layout performance optimization, i.e., layout-aware sizing. We integrate the state-of-the-art analog layout generator, MAGICAL (89), into our flow to handle layout-aware sizing. Our algorithms do not assume the pre-existence of any dataset, and we generate all training data during the optimization. We employ Bayesian Neural Networks (BNN) for modeling design performances. Bayesian Neural Networks allow error quantification, and compared to Deep Neural Networks, BNNs are shown to be effective in handling scarce datasets and preventing overfitting (27). Therefore, BNN can be trained on smaller datasets, significantly improving the practicality and scalability. We also introduce a batch-optimization framework and design space sampling strategy that is compatible with BNN modeling. Further, when optimizing the design variables based on post-layout performance, we exploit the correlation between schematic-level simulations and post-layout simulations. Our algorithm introduces a co-learning scheme that reduces the need for costly post-layout simulations and boosts efficiency even further. We compile our contributions as follows:

- We use Bayesian Neural Network-based modeling to obtain performance approximations. Different learning strategies are adapted for schematic-level sizing and post-layout performance optimization.
- We adopt a scalable sampling strategy and query the optimization batches

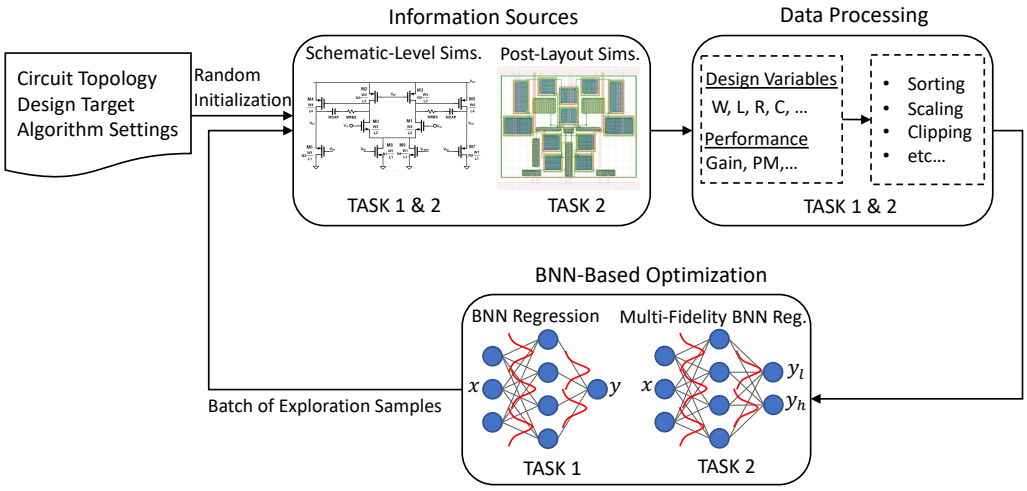


Figure 5.6: Proposed AMS Automation Framework

by utilizing a trust region and Thompson sampling.

- The post-layout sizing is handled as a multi-fidelity optimization problem, and an efficient co-learning strategy is developed.

- The efficiency of the proposed methods is demonstrated on three circuits by providing comparisons to previous state-of-the-art.

In this work, we provide solutions to two problems in Analog/Mixed-Signal design automation: schematic-level sizing automation (Task 1) and layout-aware sizing automation (Task 2). The high-level frameworks of proposed solutions to both tasks are summarized together in Figure 5.6. Section 5.4.1 introduces and elaborates on the core principles that complete

our proposed automation flow for schematic-level sizing tasks. Then, in Section 5.4.2 , we explain how to solve the post-layout performance optimization problem efficiently by transforming the BNN learning scheme.

5.4.1 Schematic-Level Sizing Automation

We propose a BNN-based sizing algorithm to optimize AMS design on schematic-level simulations. The complete flow of the proposed approach is summarized in Algorithm 3. The algorithm starts with sampling random points in the design space and simulating them via the SPICE simulator. An initial dataset for training the BNN performance model is built from these samples. Then a trust-region state is initialized before algorithm iterations start. The trust region determines the bounds of the exploration space. The following subsections will provide more details regarding the BNN modeling and trust-region search.

Our algorithm models each performance metric at each optimization iteration with an individual BNN model. Then a batch of samples is collected based on the posterior realization of points lying inside the trust region. Candidate design performance realizations are obtained using the Thompson sampling method, and the candidates are ranked based on the utility values (FoM). A batch of q points is collected, and their real performances are obtained through simulation. The new data is added to the database, and the

trust region is updated based on the real simulation outputs of the last batch.

Algorithm 3 BNN-Based Sizing Algorithm

Require: An initial sample set $\mathbf{X}^{\text{init}}$ of N_{init} designs and their evaluations $f(\mathbf{X}^{\text{init}})$

Define total population $\mathbf{X}^{\text{tot}} = \mathbf{X}^{\text{init}}$
Initialize trust-region state
Assign the solution with maximum utility
for $t = 1, 2, \ldots, t_{max}$ **do**
    Train BNN for each performance metric
    Generate r candidate points $x_1, \ldots, x_r \in \Omega$ in the trust region.
    for $b = 1, 2, \ldots, q$ **do**
        For each of the r points of the next batch, sample a realization $\{(\hat{f}(x_i), \hat{f}_1(x_i), \ldots, \hat{f}_m(x_i))^T \mid 1 \leq i \leq r\}$ from the posterior over each candidate and add a point of maximum utility to the batch.
    end for
    Simulate the q new query points and obtain specs $f(\mathbf{X}_t)$ via SPICE sims
    Update database with new designs and evaluations
    Update trust region state by comparing the current best and $f(\mathbf{X}_t)$
end for
return The design with the highest utility

5.4.1.1 Performance Modeling with Bayesian Neural Networks

We base our Bayesian Neural Network regression method on the assumption that the observed function values follow a Gaussian distribution and the probabilistic model on the observations are in the following form:

$$p(f(x) \mid x, \theta) = \mathcal{N}\left(\phi(x; \theta), \tau^{-1}\right) \tag{5.2}$$

where θ is the BNN parameters, i.e., weights and biases, $\phi(x;\theta)$ is the output of the BNN with parameters θ and τ is the noise parameter. We assign a standard Gaussian prior distribution over each element of the NN parameters, θ, and a Gamma prior over each noise precision, $p(\tau) = \text{Gam}(\tau \mid a_0, b_0)$. Let define $y_n = f(x_n)$. Given the dataset $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$, the joint probability of our model is given by

$$p(\theta, \mathcal{Y}, \tau, \mid \mathcal{X}) = \mathcal{N}(\text{vec}(\theta) \mid \mathbf{0}, \mathbf{I}) p(\tau) \prod_{n=1}^N \mathcal{N}(y_n \mid \phi(\mathbf{x}_n), \tau^{-1}) \tag{5.3}$$

where $\mathcal{X} = \{\mathbf{x}_n\}, \mathcal{Y} = \{y_n\},$ and $\text{vec}(\cdot)$ is vectorization.

Due to its unbiased, high-quality uncertainty quantification, we use Hamiltonian Monte Carlo (HMC) (59) sampling to perform posterior inference and generate samples of $\theta^i \sim p(\theta \mid \mathcal{D})$ from the posterior of BNN parameters. Then, using the samples of θ, we make a Gaussian approximation to the function value as follows:

$$\begin{aligned} \mu(f(x) \mid \mathcal{D}) &= \frac{1}{M} \sum_{i=1}^M \phi(x; \theta^i) \\ \sigma^2(f(x) \mid \mathcal{D}) &= \frac{1}{M} \sum_{i=1}^M \left(\phi(x; \theta^i) - \mu(f(x) \mid \mathcal{D})\right)^2 + \frac{1}{\tau} \end{aligned} \tag{5.4}$$

where μ is the mean and σ^2 is the variance approximation.

5.4.1.2 Trust-Region Search Engine

We follow the trust region approach introduced in (21) and confine the candidate points locally. The trust-region assigns a localized subset of the search space and proceeds in rounds. We denote the trust region by Ω. In each round, a batch of q designs in Ω are selected by the BNN algorithm and then simulated in parallel. Note that this procedure is easily extended to asynchronous batch evaluations, and we adapt asynchronous evaluation for the multi-fidelity BNN algorithm (will be discussed), where evaluation times show significant differences. The trust-region is centered around the best design explored, i.e., the design with minimum FoM where the ties are handled according to the design objective. This approach mitigates common issues of Bayesian optimization in high-dimensional settings, where popular acquisition functions fail to focus on promising regions and spread out samples due to large prediction uncertainty.

Thompson Sampling-based Exploration: We employ Thompson sampling to obtain performance approximations for untested design candidates. Thompson sampling scales to large batches at low computational cost and has shown to be as effective as the expected improvement acquisition function(21). Further, the Thompson sampling naturally extends to constrained settings which is usually the case for AMS automation. To select a point for the next batch, we sample r candidate points in Ω. Let $x_1, \ldots, x_r$

be the sampled candidate points. Then we use the predictive model given in Equation 5.4, and sample a realization $(\hat{f}_0(x_i), \hat{f}_1(x_i), ..., \hat{f}_m(x_i))^T$ for all x_i with $1 \leq i \leq r$ from the respective posterior distributions on the functions $f_0, f_1, \ldots, f_m$. Let $\hat{F} = \left\{x_i \mid \hat{f}_j(x_i) \leq 0 \text{ for } 1 \leq j \leq m\right\}$ be the set of points whose realizations are feasible. If $\hat{F} \neq \emptyset$ holds, we select an $\text{argmin}_{x \in \hat{F}} \hat{f}(x)$, i.e., the design with minimum objective. Otherwise, we select a point of minimum total violation based on the FoM definition given in Equation 2.2.

Maintaining the trust-region: We initialize a trust region as a hypercube with side length L around the maximum utility point. As the optimization progresses, we track the number of successes n_s and failures n_f since the last time the trust-region is updated. A success is when the algorithm improves the solution quality, and by construction, this point must be inside the trust region. We call it a failure when the last batch of simulated designs is worse than the current best solution. The center C of the trust region is updated as follows. If there exist feasible designs, the one with the minimum objective is assigned as the center. Otherwise, the design with minimum FoM, i.e., minimum scaled constraint violation, is chosen as the center. Therefore, the center of the trust-region is updated every time the design performance is improved. The side length of the trust region is updated as follows: if $n_s = \tau_s$ then the side length is set to $L = \min\{2L, L_{\max}\}$ and we reset $ns = 0$. If $n_f = \tau_f$, then we set $L = L/2$ and $n_f = 0$. If the side length drops below a

set threshold $L_{\min}$, we initialize a new trust region.

5.4.2 Post-Layout Performance Optimization

We start our discussion by defining the modifications necessary to automate post-layout performance-based AMS sizing. We tailor the classical sizing flow to include the post-layout effects on the performance during sizing. Instead of optimizing the design variables based on the schematic level simulations, we utilize the layout automation tool MAGICAL to modify performance evaluation steps. The suggested flow is shown in Figure 5.7. First, an automated layout is generated via MAGICAL to obtain the post-layout performance of each new design. This step is followed by parasitic extraction, and circuit simulations are run on the updated netlist with parasitic elements.

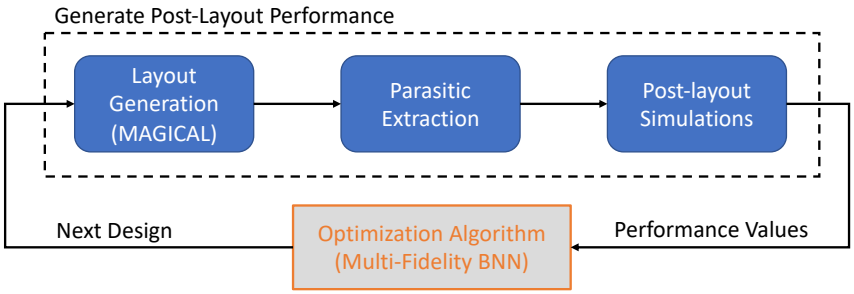


Figure 5.7: Post-Layout Performance Based Optimization

However, this new flow is much more expensive than the schematic-level

sizing task since the additional steps (layout generation, parasitic extraction, and post-layout simulations) are typically computationally expensive. Therefore, methods are sought to further increase the efficiency of the (BNN-based) optimization algorithm. As a solution, we treat this problem as a multi-fidelity problem where we have access to two different information sources for calculating circuit performance metrics. Considering that the schematic-level simulations are less accurate approximations of post-layout level simulations, we define these information sources as schematic-level simulations having the lower fidelity and post-layout simulations are the highest fidelity.

We modify the BNN architecture to capture two levels of fidelities (Figure 5.6) at the output and propose a co-learning scheme similar to multi-task BNN learning (77). The multi-fidelity BNN model has two output nodes where $\phi(x)[1]$ models the lower fidelity prediction, i.e., schematic-level performance prediction, and $\phi(x)[2]$ models the high fidelity prediction, i.e., post-layout performance prediction. Under the assumption that we have access to two levels of information sources, we denote the new dataset by $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$, where $\mathcal{D}_i = \left\{\left(\mathbf{x}_n^k, y_n^k\right)\right\}_{n=1}^{N_k}$ and the joint probability of the updated BNN model is given by:

$$p(\theta, \mathcal{Y}, \tau, \mid \mathcal{X}) = p(\theta)p\left(\tau\right)\prod_{k=1}^{2}\prod_{n=1}^{N_k}\mathcal{N}\left(y_n^k \mid \phi\left(\mathbf{x}_n^k\right)[k], \tau^{-1}\right) \quad (5.5)$$

where $p(\theta) = \mathcal{N}(\text{vec}(\theta) \mid \mathbf{0}, \mathbf{I})$ and N_k is the number training points in given

fidelity level k. The joint probability expression is a combination of the data sourced from both types of simulations; therefore, we utilize the full dataset to train multi-fidelity BNN. In this way, both fidelities are learned together, and their correlations are captured due to shared BNN parameters.

To handle the multi-fidelity problem, we adopt the following modifications to Algorithm 3:

1) We train multi-fidelity BNN models using the whole history of simulations, $\mathcal{D}$.

2) The trust-region centering and length updates are based on the post-layout simulation results, i.e., highest-level fidelity results.

3) We determine the candidate selection by modifying the work of (34) where they propose an upper-confidence-bound selection criteria for a single objective BO. We obtain Thompson sampling-based realizations for each fidelity, i.e., $\{\hat{f}_i^{(1)}(\mathrm{x}),\ \hat{f}_i^{(2)}(\mathrm{x}),\ \text{for } i = 0, 1, \ldots, m\}$ where $\{1, 2\}$ indicate the fidelity level (schematic-level simulations and post-layout level simulations) and then calculate the low fidelity and high fidelity FoM approximations, $FoM(\hat{f}^L(x))$ and $FoM(\hat{f}^H(x))$ using the corresponding realizations. The candidate selection is queried according to the following utility expression:

$$\mathcal{U}(x) = \max(FoM(\hat{f}^L(x)) - \Delta,\ FoM(\hat{f}^H(x)))$$

where Δ is the FoM difference between the samples with the best utility at

each fidelity. In this step, we take a practical approach to convert two fidelities to each other by defining a reduction term and assigning the conservative prediction as the utility value. Finally, the argmin selection is conducted on the candidate utility values to determine the next batch.

4) The current literature on multi-fidelity Bayesian optimization lacks in handling large number of constraints. Therefore, we randomly assign the fidelity (simulation type) for selected candidates and leave the fidelity selection as future work. Note that this action does not prevent us from studying the benefits of multi-fidelity handling of layout-aware sizing. However, we sacrifice potential cost-aware improvements through intelligent fidelity selection.

5.4.3 Experiments

Experiment Setup and Algorithm Settings: We run our tests using 3 different AMS circuits designed with different technologies. A Two-Stage Folded Cascode Operational Transconductance Amplifier (OTA), and a Strong-Arm Latch Comparator are designed with TSMC 180nm process and used to test schematic-level sizing algorithms. Then, we demonstrate the results for layout-aware algorithms on a Two-Stage Miller OTA. This circuit is designed in TSMC 40nm technology since the layout generator used in this work, MAGICAL, is crafted for TSMC 40nm. The schematic designs for folded

cascode OTA, strong-arm latch comparator, and miller OTA circuits are included in Figures 2.4, 2.8 and 5.2.

We run experiments to study the effectiveness of both of the proposed algorithms. First, we test for the schematic-level sizing algorithm, which is given by Algorithm 3, and we refer to our Bayesian Neural Network Based Bayesian Optimization algorithm as "BNN-BO". Then, we run tests for our post-layout performance-based sizing algorithm. Since we utilize a multi-fidelity BNN for this task, we will refer to this algorithm as "MF-BNN-BO."

We implemented several state-of-the-art baseline algorithms to compare and quantify the quality of our proposed algorithms. We selected the baseline algorithms to cover the different categories of approaches. We list the compared baseline algorithms as follows: 1) A differential evolution global optimization algorithm (DE), 2) Bayesian Optimization with weighted expected improvement (BO) (57), and 3) RL-based sizing algorithm, DNN-Opt (6). All algorithms are implemented using Python. We implemented DNN-Opt via PyTorch (61), Bayesian Optimization algorithm is implemented using BoTorch (3) package and BNN-BO and MF-BNN-BO are implemented using PyTorch and Hamiltorch (15) packages.

We configured BNN-BO and MF-BNN-BO to evaluate a parallel batch of $q = 8$ designs. For fairness, DNN-Opt and BO are also configured to do parallel evaluations. Both our algorithms use 200 HMC samples to train BNN

models. Trust-region is initiated with $L = 0.8$ and L_{min} and L_{max} are chosen to be 0.5^4 and 1.6, respectively. Failure and success tolerances as chosen as $n_f = 2$ and $n_s = 3$. All experiments are run on the same machine using the CPU for training learning (DNN and BNN) models. During experiments, the model-based algorithms BO, DNN-Opt, and BNN-BO are run until exploring 500 designs, and DE is run for 5000 new samples.

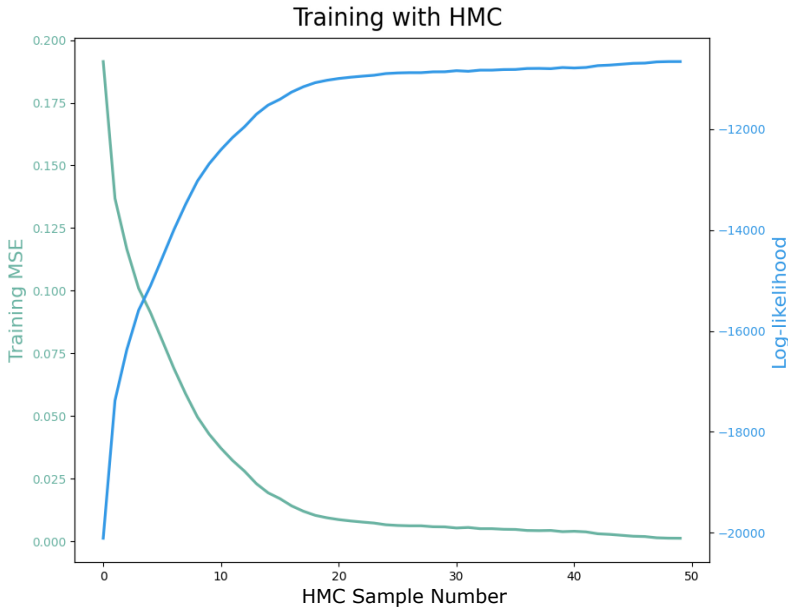


Figure 5.8: Folded Cascode OTA Power Modeling

Schematic-Level Sizing Automation: We tested our algorithm, BNN-BO, and other baseline algorithms on two circuits: two-stage folded cascode ota and strong-arm latch comparator. All transistors in both designs are

parameterized for optimization. The Folded Cascode OTA has 20 independent design variables, and the Strong-Arm Latch Comparator has 13 independent design variables after respecting the symmetry constraints. The parameterized device sizes include: transistor lengths & widths, capacitor values, and multipliers (integer-valued).

The schematic-level constrained sizing problem for Folded Cascode OTA is defined as follows:

$$
\begin{aligned}
&\text{minimize Power} \\
&\text{s.t.} \quad \text{DC Gain} > 60 \ \text{dB} \qquad \text{Settling Time} < 30 \ \text{ns} \\
&\qquad \text{CMRR} > 80 \ \text{dB} \qquad \text{Saturation Margin} > 50 \ \text{mV} \\
&\qquad \text{PSRR} > 80 \ \text{dB} \qquad \text{Unity Gain Freq.} > 30 \ \text{MHz} \\
&\qquad \text{Out. Swing} > 2.4 \ \text{V} \quad \text{Out. Noise} < 30 \ \text{mV}_{\text{rms}} \\
&\qquad \text{Static error} < 0.1 \qquad \text{Phase Margin} > 60 \ \text{deg.}
\end{aligned}
\tag{5.6}
$$

The schematic-level constrained sizing problem for Strong-Arm Latch Comparator is defined as follows:

$$
\begin{aligned}
&\text{minimize Power} \\
&\text{s.t.} \quad \text{Set Delay} < 10 \ \text{ns} \\
&\qquad \text{Reset Delay} < 6.5 \ \text{ns} \\
&\qquad \text{Input-referred Noise} < 50 \ \mu\text{Vrms} \\
&\qquad \text{Differential Reset Voltage} < 1 \ \mu\text{V} \\
&\qquad \text{Differential Set Voltage} > 1.195 \ \text{V} \\
&\qquad \text{Positive-Integration Node Reset Voltage} < 60 \ \mu\text{V} \\
&\qquad \text{Negative-Integration Node Reset Voltage} < 60 \ \mu\text{V} \\
&\qquad \text{Positive-Output Node Reset Voltage} < 0.35 \ \mu\text{V} \\
&\qquad \text{Negative-Output Node Reset Voltage} < 0.35 \ \mu\text{V}.
\end{aligned}
\tag{5.7}
$$

We show the accuracy of the BNN modeling by demonstrating the training metrics. Training Mean Squared Error (MSE) and the logarithmic likelihood of the fitted model are given in Figure 5.8. Collecting new HMC samples from the posterior increases the likelihood and reduces the training

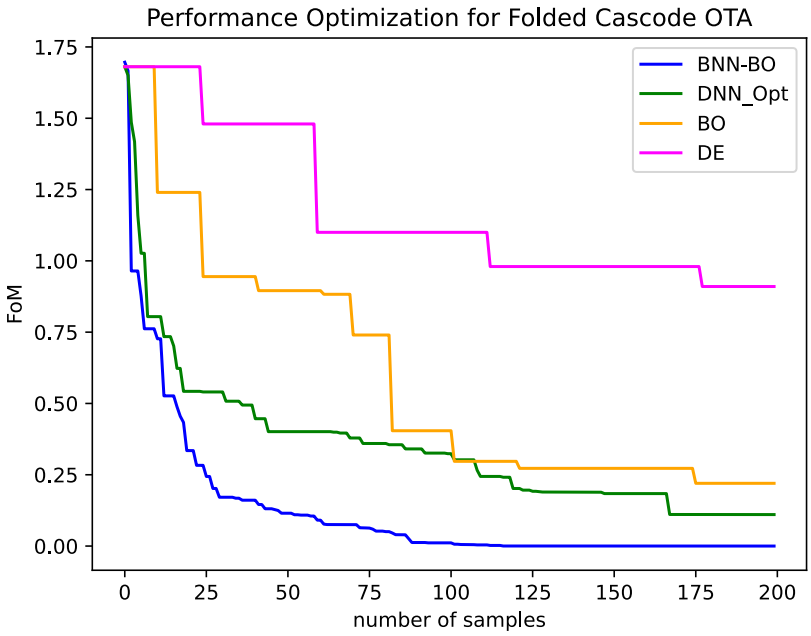


Figure 5.9: Folded Cascode Optimization

error. We observed very similar training schemes for all other circuits and performance metrics.

We repeat all experiments 10 times to account for the randomization involved in tested algorithms. The statistical results of our tests are shown in Table 5.5. Testing on both circuits suggests that BNN-BO can achieve feasible solutions in all runs, and it uses the smallest number of simulations to achieve this. Compared to Differential Evolution (DE), BNN-BO can find feasible solutions using up to 40x less number of simulations. Compared to the closest baseline algorithm, DNN-Opt, BNN-BO reduces the simulation time for finding similar results by up to 56%, proving its high efficiency. Considering that BNN-BO requires more time for modeling, the total time efficiency for finding feasible results becomes up to 51%. It is also demonstrated in Table 5.5 that, on average, the final design proposed by BNN-BO draws up to 40% less power. The only disadvantage of BNN-BO to DNN-Opt is the modeling time

Table 5.5: Schematic-Level Sizing Optimization Statistics

Circuit Name	Folded Cascode OTA				Strong-Arm Latch Comparator			
Algorithm	DE	BO	DNN-Opt	**BNN-BO**	DE	BO	DNN-Opt	**BNN-BO**
success rate	10/10	7/10	10/10	**10/10**	7/10	1/10	9/10	**10/10**
# of simulations	3200	340	151	**82**	2800	>500	154	**68**
Min power (mW)	0.75	0.88	0.64	**0.60**	3.02	3.67	**2.45**	2.5
Max power (mW)	1.53	1.43	0.8	**0.75**	4.1	3.67	2.66	**2.55**
Mean pow. (mW)	1.14	1.19	0.72	**0.69**	3.44	3.67	2.54	**2.52**
Modeling time (h)	NA	30	**0.6**	1.5	NA	17	**0.3**	0.7
Simulation time (h)	54	2.7	2.7	2.7	72	3.6	3.6	3.6
Total runtime (h)	54	32.7	**3.3**	4.2	72	20.6	**3.9**	4.3

as DNN-Opt maintains a single DNN model to approximate all performance metrics. Note that all reported times consider the full simulation budget (500 new samples). Therefore, although it takes longer time for BNN-BO to do a single iteration, the required real-time for BNN-BO to find a feasible solution is still smaller than other approaches.

In addition to experiment statistics, we further include the FoM convergence curves of both tests in Figure 5.9 and Figure 5.10. The y-axis in the graphs represents the total constraint violation; therefore, FoM=0 represents a feasible solution. We observe that, compared to DNN-Opt, BNN-BO has 65% and 33% smaller area under the curve for Folded Cascode OTA and SA Latch Comparator, respectively.

Layout-Aware Design Automation: In order to demonstrate the importance of layout effects on the final performance, we perform experiments on a Miller OTA circuit designed in 40nm technology (Fig. 5.2). The optimization problem has 17 independent design variables and the optimization problem is defined as follows:

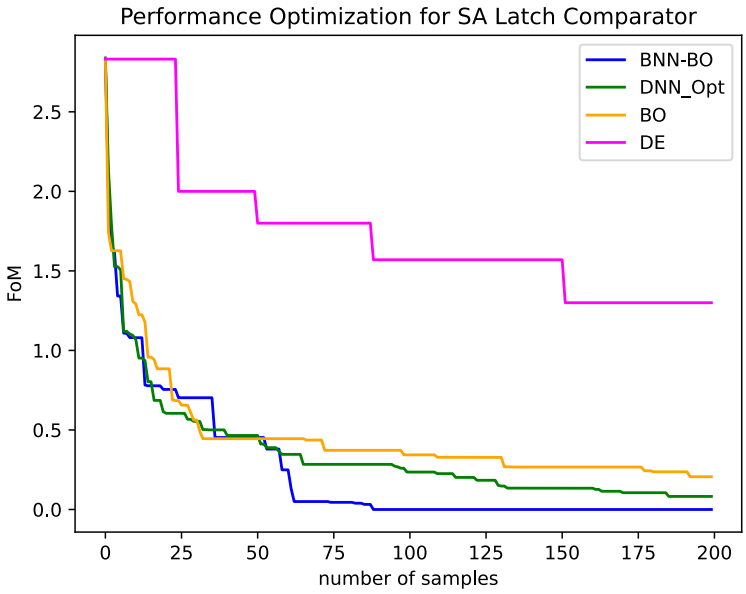


Figure 5.10: Strong-Arm Latch Comparator Optimization

$$
\begin{array}{lll}
\text{minimize Power} & & \\
\text{s.t.} & \text{DC Gain} > 45 \text{ dB} & \text{Settling Time} < 100 \text{ ns} \\
& \text{CMRR} > 55 \text{ dB} & \text{Saturation Margins} > 50 \text{ mV} \\
& \text{PSRR} > 55 \text{ dB} & \text{Unity Gain BW.} > 40 \text{ MHz} \\
& \text{Out. Swing} > 1 \text{ V} & \text{RMS Noise} < 400 \text{ uV}_{\text{rms}} \\
& \text{Static error} < \%2 & \text{Phase Margin} > 60 \text{ deg.}
\end{array}
\tag{5.8}
$$

Obtaining the post-layout performance of the Miller OTA is around 9 times more expensive than obtaining the schematic-level performance. Therefore this experiment is to prove the efficiency by utilizing multiple information sources. We initialize all algorithms with 50 high-fidelity random samples, and MF-BNN-BO has additional 50 samples from low-fidelity source (schematic-level simulations). We demonstrate the FoM evolution for the rest of the optimization steps in Figure 5.11. We observe that our Multi-Fidelity BNN

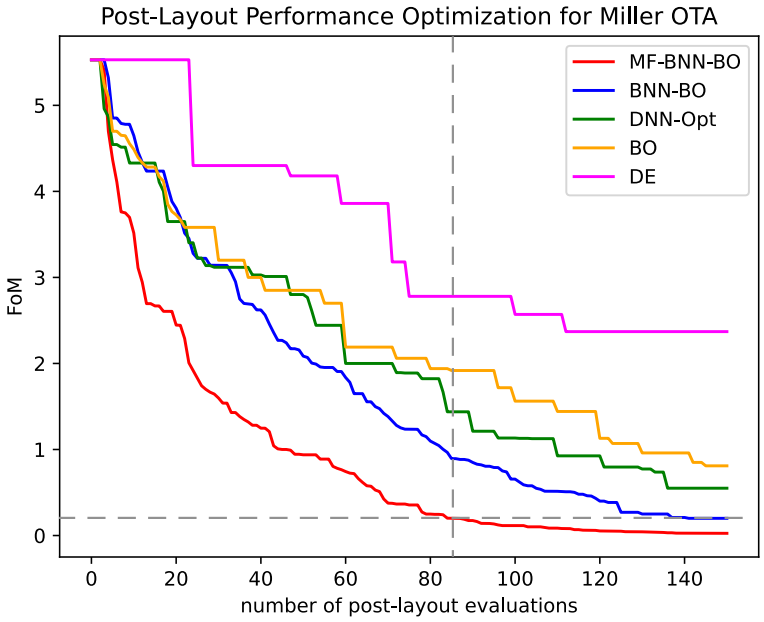


Figure 5.11: Miller OTA Post-Layout Performance Optimization

algorithm provides even more efficiency compared to already efficient BNN-BO. Our analysis shows that the area under the curve is 45% smaller for MF-BNN-BO compared to BNN-BO. Further, we observe that BNN-BO's average best solution after 150 high-fidelity iterations is surpassed by MF-BNN-BO only using 84 simulations. This also implies close to 45% improved efficiency due to utilizing correlations between the schematic-level evaluations and post-layout level evaluations. Note that there is an equal number of schematic-level simulations while running MF-BNN-BO that are not reflected in Figure 5.11. Considering these simulations, the time efficiency is slightly reduced to around 38%. This efficiency figure serves as a lower-bound since we leave the improvements on fidelity selection as a future work.

5.5 Summary

In this chapter, we first presented a case study to analyze and quantify the effects of post-layout parasitics. We show that ignoring the layout-induced effects significantly lowers end-design quality if the optimization is solely conducted on schematic-level simulations. Further, we presented Bayesian Neural Network-based solutions for schematic-level analog sizing automation and post-layout performance optimization. We targeted the scalability issue of the learning-based automation methods and provided a sample efficient optimization flow. We demonstrated the efficiency of the proposed approaches on academic benchmarks. Compared to the state-of-the-art, we improved the sizing automation efficiency by up to 55%. The Multi Fidelity BNN algorithm analysis proved that utilizing cheaper (schematic-level) simulations reduces the need for expensive (post-layout) simulations considerably, further boosting the efficiency.

Chapter 6: Conclusion and Future Directions

In this dissertation, we proposed various techniques and algorithms to tackle problems in the field of AMS sizing problem. First, a new methodology of ML-Assisted global optimization method is presented. This novel approach formalizes an overall framework for improving the efficiency of evolutionary search algorithms. It includes elements for utilizing small datasets for performance modeling, which is typically a bottleneck for AMS data. We demonstrate that ANN-based modeling could significantly reduce the required simulations compared to conventional approaches. Second, a reinforcement-learning-inspired optimization scheme is developed to target the sizing problem. We explained how the sizing problem can be mapped into an RL problem. Further, we proposed a recipe to extend our algorithm to large-scale industrial circuits. The overall algorithm's efficiency is demonstrated on various academic and industrial circuits. Third, we introduced a solution for AMS circuits with varying simulations. Specifically, we developed a theory to make intermediate decisions regarding design quality by assessing the performance metrics obtained from relatively cheaper simulations. Our gaussian process based rank prediction model is supported by a theory-based conditioning to determine when to run expensive simulations for a particular design. Fur-

ther, we improved the overall real-time efficiency of this work by introducing an asynchronous parallel framework. The overall algorithm significantly improved real-time efficiency compared to the state-of-the-art baseline. Finally, we investigated the effect of layout-induced parasitics and proposed a novel approach to efficiently optimize the performance of AMS blocks in the presence of layout effects. We first demonstrated the efficiency of Bayesian Neural Networks on schematic-level sizing problems. Then, we adapted a co-learning strategy to model schematic-level and post-layout data together for efficient learning. We explained how to train multi-fidelity Bayesian Neural Networks to solve post-layout performance optimization problems in an efficient way. In summary, we have introduced several solutions to size AMS circuits, both based on schematic-level simulations and post-layout simulations.

Despite our efforts to bring automated solutions for various aspects and cases of AMS sizing, many related problems are potential research topics. The topics we list below can serve as complementary research problems that would help AMS sizing automation be complete.

- A major contribution to AMS design automation would be integrating sizing and layout generation phases. Currently, most tools isolate these phases or handle only one at a time. However, there is a large coupling between the two phases. One chapter in this dissertation is dedicated

to conducting layout-aware AMS sizing. However, we did not allow any feedback between sizing and layout generation. The optimization algorithm used the layout generator as a black-box in our work. We believe using the history of sizing decisions and the post-layout performance data as input to the layout generator could significantly improve the final design quality.

- There are several factors in AMS design that causes the manufactured chip performance to deviate from the nominally simulated one. Specifically, the final chip's performance can be significantly impacted due to process, voltage, and temperature (PVT) variations. Therefore, robustness against these variations is crucial for real-life conditions. Designers generally check the design performance under different temperature corners and varying voltage sources. Also, they utilize Monte Carlo simulations for statistical evaluations of process variations. Following this practice, AMS sizing automation could also propose solutions that consider PVT variations. In this case, the problem formulation may change. For example, to obtain certain performance across corners, a minmax optimization is conducted over different design corners.

- One major and common issue with optimization algorithms used for AMS sizing is the curse of dimensionality. Most algorithms' performance

is significantly reduced when the input space becomes larger. Our proposal for this issue was to utilize sensitivity between design variables and performance metrics. However, our approach assumes the existence of these sensitivities and an initial working design where the circuit is normalized at. New methods easing these requisites could result in more computationally efficient algorithms.

- AMS design automation could significantly benefit from Transfer Learning (RL) techniques. Considering that AMS simulations can be very expensive, generating analog data is a tedious process. Therefore, utilizing existing data could help greatly in reducing total time of data preparation and training for ML models. We think that TL could especially be useful for the cases where the data dimensionality is preserved. For example, the topology is generally unchanged when a particular design is ported between design processes. In this case, the input and output features between the original and ported designs are highly overlapped. Another case is when a constant topology is needed to be reconfigured for a new application. In this case, the existing data can be used directly to train on the new problem with the updated spec list.

In summary, there has been a great proof of concept for AMS IC sizing, but many aspects must be considered to better capture the full design closure.

The developed algorithms can help mitigate issues with AMS design, improve circuit performance, and reduce human effort. With future improvement, AMS design optimization tools could improve design quality, enhance productivity and reduce the design cycle.

Works Cited

[1] M. O. Akinsolu, B. Liu, V. Grout, P. I. Lazaridis, M. E. Mognaschi, and P. Di Barba, "A parallel surrogate model assisted evolutionary algorithm for electromagnetic design optimization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 2, pp. 93–105, 2019.

[2] G. Alpaydin, S. Balkir, and G. Dundar, "An evolutionary approach to automatic synthesis of high-performance analog integrated circuits," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, pp. 240–252, 2003.

[3] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, "BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization," in *Advances in Neural Information Processing Systems 33*, 2020.

[4] M. Barros, J. Guilherme, and N. Horta, "Analog circuits optimization based on evolutionary computation techniques," *Integration*, vol. 43, no. 1, pp. 136–155, 2010.

[5] A. Budak, M. Gandara, W. Shi, D. Pan, N. Sun, and B. Liu, "An efficient analog circuit sizing method based on machine learning assisted

global optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[6] A. F. Budak, P. Bhansali, B. Liu, N. Sun, D. Z. Pan, and C. V. Kashyap, "DNN-Opt an RL inspired optimization for analog circuit sizing using deep neural networks," in *ACM/IEEE Design Automation Conference (DAC)*, 2021.

[7] A. F. Budak, Z. Jiang, K. Zhu, A. Mirhoseini, A. Goldie, and D. Z. Pan, "Reinforcement learning for electronic design automation: Case studies and perspectives: (invited paper)," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2022.

[8] A. F. Budak, D. Smart, B. Swahn, and D. Z. Pan, "APOSTLE: asynchronously parallel optimization for sizing analog transistors using dnn learning," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2023.

[9] A. F. Budak, S. Zhang, M. Liu, W. Shi, K. Zhu, and D. Z. Pan, "Machine learning for analog circuit sizing," in *Machine Learning for Analog Circuit Sizing*, H. Ren and J. Hu, Eds. Springer, 2022, pp. 307–335.

[10] A. F. Budak, K. Zhu, H. Chen, S. Poddar, L. Zhao, Y. Jia, and D. Z. Pan, "Joint optimization of sizing and layout for ams designs: Challenges

and opportunities," in *Proceedings of the 2023 International Symposium on Physical Design*, ser. ISPD '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 84–92.

[11] A. F. Budak, K. Zhu, and D. Z. Pan, "Practical layout-aware analog/mixed-signal design automation with bayesian neural networks," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2023.

[12] H. Chen, M. Liu, X. Tang, K. Zhu, A. Mukherjee, N. Sun, and D. Z. Pan, "MAGICAL 1.0: An open-source fully-automated ams layout synthesis framework verified with a 40-nm 1 GS/s $\Delta\Sigma$ ADC," in *Custom Integrated Circuits Conference (CICC)*, 2021.

[13] H. Chen, M. Liu, X. Tang, K. Zhu, N. Sun, and D. Z. Pan, "Challenges and opportunities toward fully automated analog layout design," *Journal of Semiconductors*, vol. 41, no. 20070021, p. 111407, 2020.

[14] H. Chen, K. Zhu, M. Liu, X. Tang, N. Sun, and D. Z. Pan, "Toward silicon-proven detailed routing for analog and mixed signal circuit," in *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020.

[15] A. D. Cobb and B. Jalaian, "Scaling hamiltonian monte carlo inference for bayesian neural networks with symmetric splitting," *Uncertainty in*

Artificial Intelligence, 2021.

[16] J. Crossley, A. Puggelli, H.-P. Le, B. Yang, R. Nancollas, K. Jung, L. Kong, N. Narevsky, Y. Lu, N. Sutardja, E. J. An, A. L. Sangiovanni-Vincentelli, and E. Alon, "BAG: A designer-oriented integrated framework for the development of ams circuit generators," in *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2013.

[17] M. d. Hershenson, S. P. Boyd, and T. H. Lee, "Optimal design of a cmos op-amp via geometric programming," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2001.

[18] M. Dehbashian and M. Maymandi-Nejad, "A new hybrid algorithm for analog ics optimization based on the shrinking circles technique," *Integration*, vol. 56, pp. 148–166, 2017.

[19] N. Drira, M. Kotti, M. Fakhfakh, P. Siarry, and E. Tlelo-Cuautle, "Convergence rates of the efficient global optimization algorithm for improving the design of analog circuits," *Analog Integrated Circuits and Signal Processing*, pp. 1–20, 2020.

[20] M. Emmerich, K. Giannakoglou, and B. Naujoks, "Single-and multiobjective evolutionary optimization assisted by Gaussian random field metamodels," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4,

pp. 421–439, 2006.

[21] D. Eriksson and M. Poloczek, "Scalable constrained bayesian optimization," in *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, A. B. 0001 and K. Fukumizu, Eds., vol. 130. PMLR, 2021.

[22] M. Fakhfakh, E. Tlelo-Cuautle, and M. H. Fino, *Performance Optimization Techniques in Analog, Mixed-Signal, and Radio-Frequency Circuit Design*. IGI Global, 2014.

[23] Z. Gao, J. Tao, F. Yang, Y. Su, D. Zhou, and X. Zeng, "Efficient performance trade-off modeling for analog circuit based on bayesian neural network," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019, pp. 1–8.

[24] A. Garbaya, M. Kotti, N. Drira, M. Fakhfakh, E. Tlelo-Cuautle, and P. Siarry, "An rbf-pso technique for the rapid optimization of (cmos) analog circuits," in *2018 7th International Conference on Modern Circuits and Systems Technologies (MOCAST)*. IEEE, 2018, pp. 1–4.

[25] A. Garbaya, M. Kotti, M. Fakhfakh, and E. Tlelo-Cuautle, "Surrogate assisted optimization for low-voltage low-power circuit design," *Journal*

of Low Power Electronics and Applications, vol. 10, no. 2, p. 20, 2020.

[26] J. R. Gardner, M. J. Kusner, Z. E. Xu, K. Q. Weinberger, and J. P. Cunningham, "Bayesian optimization with inequality constraints." in *ICML*, vol. 2014, 2014, pp. 937–945.

[27] E. Goan and C. Fookes, "Bayesian neural networks: An introduction and survey," in *Case Studies in Applied Bayesian Data Science*. Springer International Publishing, 2020, pp. 45–87.

[28] K. Hakhamaneshi, N. Werblun, P. Abbeel, and V. Stojanović, "BagNet: Berkeley analog generator with layout optimizer boosted with deep neural networks," in *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2019.

[29] B. He, S. Zhang, Y. Wang, T. Gao, F. Yang, C. Yan, D. Zhou, Z. Bi, and X. Zeng, "A batched bayesian optimization approach for analog circuit synthesis via multi-fidelity modeling," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 2, pp. 347–359, 2023.

[30] B. He, S. Zhang, F. Yang, C. Yan, D. Zhou, and X. Zeng, "An efficient bayesian optimization approach for analog circuit synthesis via sparse

gaussian process modeling," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 67–72.

[31] G. Huang, J. Hu, Y. He, J. Liu, M. Ma, Z. Shen, J. Wu, Y. Xu, H. Zhang, K. Zhong, X. Ning, Y. Ma, H. Yang, B. Yu, H. Yang, and Y. Wang, "Machine learning for electronic design automation: A survey," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 26, no. 5, Jun. 2021.

[32] J. Huang, F. Yang, C. Yan, D. Zhou, and X. Zeng, "A robust batch bayesian optimization for analog circuit synthesis via local penalization," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2021.

[33] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[34] K. Kandasamy, G. Dasarathy, J. B. Oliva, J. Schneider, and B. Poczos, "Gaussian process bandit optimisation with multi-fidelity evaluations," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[35] D. Kingma and L. Ba, "Adam: A method for stochastic optimization," 2015.

[36] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *SIAM Journal on Control and Optimization*. MIT Press, 2000.

[37] M. Kotti, M. Fakhfakh, and E. Tlelo-Cuautle, "Kriging metamodeling-assisted multi-objective optimization of cmos current conveyors," in *2018 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. IEEE, 2018, pp. 293–296.

[38] K. Kunal, M. Madhusudan, A. K. Sharma, W. Xu, S. M. Burns, R. Harjani, J. Hu, D. A. Kirkpatrick, and S. S. Sapatnekar, "ALIGN: Open-source analog layout automation from the ground up," in *ACM/IEEE Design Automation Conference (DAC)*, 2019.

[39] Y. Li, Y. Wang, Y. Li, R. Zhou, and Z. Lin, "An artificial neural network assisted optimization system for analog design space exploration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019.

[40] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learn-

ing." 2016.

[41] B. Liu, D. Zhao, P. Reynaert, and G. G. E. Gielen, "Gaspad: A general and efficient mm-wave integrated circuit synthesis method based on surrogate model assisted evolutionary algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 2, pp. 169–182, 2014.

[42] B. Liu, Q. Chen, Q. Zhang, G. Gielen, and V. Grout, "Behavioral study of the surrogate model-aware evolutionary search framework," in *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 715–722.

[43] B. Liu, N. Deferm, D. Zhao, P. Reynaert, and G. G. Gielen, "An efficient high-frequency linear rf amplifier synthesis method based on evolutionary computation and machine learning techniques," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 7, pp. 981–993, 2012.

[44] B. Liu, F. V. Fernández, G. Gielen, R. Castro-López, and E. Roca, "A memetic approach to the automatic design of high-performance analog integrated circuits," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 14, no. 3, pp. 1–24, 2009.

[45] B. Liu, G. Gielen, and F. V. Fernández, "Automated design of analog and high-frequency circuits," *A computational intelligence approach, Springer, Berlin, Heidelberg*, pp. 978–3, 2014.

[46] B. Liu, G. Gielen, and F. V. Fernndez, *Automated Design of Analog and High-frequency Circuits: A Computational Intelligence Approach*. Springer, 2013.

[47] B. Liu, Q. Zhang, and G. G. Gielen, "A gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 180–192, 2013.

[48] B. Liu, D. Zhao, P. Reynaert, and G. G. Gielen, "Synthesis of integrated passive components for high-frequency rf ics based on evolutionary computation and machine learning techniques," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 10, pp. 1458–1468, 2011.

[49] J. Liu, S. Su, M. Madhusudan, M. Hassanpourghadi, S. Saunders, Q. Zhang, R. Rasul, Y. Li, J. Hu, A. K. Sharma, S. S. Sapatnekar, R. Harjani, A. Levi, S. Gupta, and M. S.-W. Chen, "From specification to silicon: Towards analog/mixed-signal design automation using surrogate nn mod-

els with transfer learning," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021, pp. 1–9.

[50] M. Liu, W. Li, K. Zhu, B. Xu, Y. Lin, L. Shen, X. Tang, N. Sun, and D. Z. Pan, "S^{3}DET: Detecting system symmetry constraints for analog circuits with graph similarity," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020.

[51] M. Liu, W. J. Turner, G. F. Kokai, B. Khailany, D. Z. Pan, and H. Ren, "Parasitic-aware analog circuit sizing with graph neural networks and bayesian optimization," in *Design, Automation and Test in Europe Conference and Exhibition*, 2021.

[52] M. Liu, K. Zhu, J. Gu, L. Shen, X. Tang, N. Sun, and D. Z. Pan, "Towards decrypting the art of analog layout: Placement quality prediction via transfer learning," in *Design, Automation and Test in Europe Conference and Exhibition*, 2020.

[53] N. Lourenço, E. Afacan, R. Martins, F. Passos, A. Canelas, R. Póvoa, N. Horta, and G. Dundar, "Using polynomial regression and artificial neural networks for reusable analog ic sizing," in *2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. IEEE, 2019, pp. 13–16.

[54] N. Lourenço, R. Martins, A. Canelas, R. Póvoa, and N. Horta, "Aida: Layout-aware analog circuit-level sizing with in-loop layout generation," *Integration*, vol. 55, pp. 316–329, 2016.

[55] W. Lyu *et al.*, "Batch bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design," in *ICML*, 2018.

[56] W. Lyu, P. Xue, F. Yang, C. Yan, Z. Hong, X. Zeng, and D. Zhou, "An efficient bayesian optimization approach for automated optimization of analog circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 6, pp. 1954–1967, 2017.

[57] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Multi-objective bayesian optimization for analog/rf circuit synthesis," in *ACM/IEEE Design Automation Conference (DAC)*, 2018.

[58] A. Mukherjee, M. Gandara, X. Yang, L. Shen, X. Tang, C.-K. Hsu, and N. Sun, "A 74.5-db dynamic range 10-mhz bw ct-$\delta\sigma$ adc with distributed-input vco and embedded capacitive-π network in 40-nm cmos," *IEEE Journal of Solid-State Circuits*, 2020.

[59] R. M. Neal, "MCMC using Hamiltonian dynamics," *Handbook of Markov Chain Monte Carlo*, vol. 54, pp. 113–162, 2010.

[60] J. M. Parr, A. J. Keane, A. I. Forrester, and C. M. Holden, "Infill sampling criteria for surrogate-based optimization with constraint handling," *Engineering Optimization*, vol. 44, no. 10, pp. 1147–1166, 2012.

[61] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[62] K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[63] S. S. Rao, *Engineering optimization: theory and practice*. John Wiley & Sons, 2019.

[64] M. Rapp, H. Amrouch, Y. Lin, B. Yu, D. Z. Pan, M. Wolf, and J. Henkel, "Mlcad: A survey of research in machine learning for cad keynote paper," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2021.

[65] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[66] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, 2006, vol. 1.

[67] H. Ren, G. F. Kokai, W. J. Turner, and T.-S. Ku, "ParaGraph: Layout parasitics and device parameter prediction using graph neural networks," in *ACM/IEEE Design Automation Conference (DAC)*, July 2020.

[68] H. Ren, S. Godil, B. Khailany, R. Kirby, H. Liao, S. Nath, J. Raiman, and R. Roy, "Optimizing vlsi implementation with reinforcement learning," in *2021 International Conference On Computer-Aided Design (ICCAD)*. IEEE/ACM, 2021.

[69] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolić, "AutoCkt: deep reinforcement learning of analog circuit designs," *Design, Automation and Test in Europe Conference and Exhibition*, 2020.

[70] K. Settaluri, Z. Liu, R. Khurana, A. Mirhaj, R. Jain, and B. Nikolic, "Automated design of analog circuits using reinforcement learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[71] K. Shao, Z. Tang, Y. Zhu, N. Li, and D. Zhao, "A survey of deep reinforcement learning in video games," *arXiv preprint arXiv:1912.10944*, 2019.

[72] L. Shen, N. Lu, and N. Sun, "A 1v 0.25 uw inverter-stacking amplifier with 1.07 noise efficiency factor," in *2017 Symposium on VLSI Circuits*. IEEE, 2017, pp. C140–C141.

[73] B. Shook, P. Bhansali, C. Kashyap, C. Amin, and S. Joshi, "Mlparest: Machine learning based parasitic estimation for custom circuit design," in *ACM/IEEE Design Automation Conference (DAC)*, 2020.

[74] B. Shook, P. Bhansali, C. Kashyap, C. Amin, and S. Joshi, "Mlparest: Machine learning based parasitic estimation for custom circuit design," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[75] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[76] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *NIPS*, vol. 25, 2012.

[77] J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter, "Bayesian optimization with robust bayesian neural networks," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[78] M. Stein, "Large sample properties of simulations using Latin hypercube sampling," *Technometrics*, pp. 143–151, 1987.

[79] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[80] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[81] X. Tang, L. Shen, B. Kasap, X. Yang, W. Shi, A. Mukherjee, D. Z. Pan, and N. Sun, "An energy-efficient comparator with dynamic floating inverter amplifier," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 4, pp. 1011–1022, 2020.

[82] E. Tlelo-Cuautle, P. R. Castañeda-Aviña, R. Trejo-Guerra, and V. H. Carbajal-Gómez, "Design of a wide-band voltage-controlled ring oscillator implemented in 180 nm cmos technology," *Electronics*, vol. 8, no. 10, p. 1156, 2019.

[83] E. Tlelo-Cuautle, M. A. Valencia-Ponce, and L. G. de la Fraga, "Sizing cmos amplifiers by pso and mol to improve dc operating point conditions," *Electronics*, vol. 9, no. 6, p. 1027, 2020.

[84] R. Turner and D. Eriksson, "Bayesmark," https://github.com/uber/bayesmark, 2020.

[85] A. Vural *et al.*, "Analog circuit sizing via swarm intelligence," *AEU - International Journal of Electronics and Communications*, 2012.

[86] H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H. Lee, and S. Han, "GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," *ACM/IEEE Design Automation Conference (DAC)*, 2020.

[87] Y. Wang, P. D. Franzon, D. Smart, and B. Swahn, "Multi-fidelity surrogate-based optimization for electromagnetic simulation acceleration," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, aug 2020.

[88] P. D. Wasserman, *Advanced methods in neural computing*. John Wiley & Sons, Inc., 1993.

[89] B. Xu, K. Zhu, M. Liu, Y. Lin, S. Li, X. Tang, N. Sun, and D. Z. Pan, "MAGICAL: Toward fully automated analog IC layout leveraging human

and machine intelligence," in *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2019.

[90] K. Yang *et al.*, "Trust-region method with deep reinforcement learning in analog design space exploration," in *ACM/IEEE Design Automation Conference (DAC)*, 2021.

[91] Y. Yang, H. Zhu, Z. Bi, C. Yan, D. Zhou, Y. Su, and X. Zeng, "Smart-msp: a self-adaptive multiple starting point optimization approach for analog circuit synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 3, pp. 531–544, 2017.

[92] S. Zhang *et al.*, "An efficient asynchronous batch bayesian optimization approach for analog circuit synthesis," in *ACM/IEEE Design Automation Conference (DAC)*, 2020.

[93] S. Zhang, W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng, "Bayesian optimization approach for analog circuit synthesis using neural network," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1463–1468.

[94] K. Zhu, H. Chen, M. Liu, and D. Z. Pan, "Tutorial and perspectives on MAGICAL: A silicon-proven open-source analog IC layout system," *The IEEE Transactions on Circuits and Systems—II*, 2022.

[95] K. Zhu, H. Chen, M. Liu, X. Tang, N. Sun, and D. Z. Pan, "Effective analog/mixed-signal circuit placement considering system signal flow," in *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020.

[96] K. Zhu, H. Chen, W. Turner, G. Kokai, P.-H. Wei, D. Z. Pan, and H. Ren, "TAG: Learning circuit spatial embedding from layouts," in *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2022.

Vita

Ahmet Faruk Budak was born in Istanbul, Turkey, and raised in Bursa, Turkey. He received his B.S. degree from the Department of Electrical & Electronics Engineering, Bogazici University, with High Honors degree. He double majored in Physics. He was a recipient of TUBITAK scholarship during his undergraduate studies. He joined The University of Texas Department of Electrical and Computer Engineering, Austin, TX, USA, in Fall 2018 where he is currently pursuing his Ph.D. degree. From July 2020 to September 2020, he interned at Intel Corp. From May 2021 to December 2022, he interned at Analog Devices Inc.

His current research interests include machine learning applications for analog/mixed-signal design automation, analog circuit sizing, layout-aware design parameter optimization, and robust optimization. He is a recipient of ADI Outstanding Student Designer Award in 2021. His work is nominated for Best Paper Award at Design Automation Conference (DAC) 2021.

Address: ahmetfarukbudak@utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.