

BagNet: Berkeley Analog Generator with Layout Optimizer Boosted with Deep Neural Networks

kourosh hakhamaneshi



Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-25

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-25.html>

May 6, 2019

Copyright © 2019, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

**BagNet: Berkeley Analog Generator with Layout Optimizer Boosted with
Deep Neural Networks**

by

Kourosh Hakhamaneshi

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

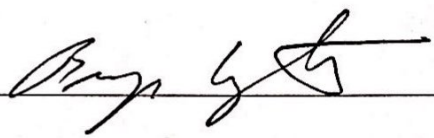

University of California, Berkeley

Committee in charge:

Professor Vladimir Stojanovic, Chair
Professor Pieter Abbeel

Spring 2019

The thesis of Kourosh Hakhamaneshi, titled BagNet: Berkeley Analog Generator with Layout Optimizer Boosted with Deep Neural Networks, is approved:

Chair		Date	<u>5/3/19</u>
		Date	<u>5/5/2019</u>
		Date	_____

University of California, Berkeley

**BagNet: Berkeley Analog Generator with Layout Optimizer Boosted with
Deep Neural Networks**

Copyright 2019
by
Kourosh Hakhamaneshi

Abstract

BagNet: Berkeley Analog Generator with Layout Optimizer Boosted with Deep Neural Networks

by

Kouros Hakhmaneshi

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Vladimir Stojanovic, Chair

In this project report we demonstrate a new Deep Learning approach for automating layout-level design of analog circuits. Post-layout simulation is computationally expensive, making traditional optimization-based design that relies directly on such simulation impractical for any but the very simplest designs. We show that deep neural nets can be trained to (mostly) substitute for the expensive simulation in the inner optimization, making it practical to solve challenging layout design problems. At the core are an evolutionary optimization approach and a neural net that predicts the performance of members of the new generation compared to the old one. The optimizer periodically runs the (much more expensive) simulation to further refine the prediction of the neural net as the population evolves, which ensures the neural net is sufficiently precise in the currently relevant part of the design space. Compared to relying on simulation for every evaluation, our approach is several orders of magnitude more efficient, enabling significantly more complex designs. Indeed, we demonstrate that our framework can efficiently explore the design space of a variety of complex circuits, including a two stage OpAmp with positive feedback and an optical links analog front end, given high level specifications and propose satisfying solutions in the end.

Contents

Contents	i
List of Figures	iii
List of Tables	iv
1 Introduction	1
1.1 Integrated Circuit Market Overview	1
1.2 Analog IC Design Flow	1
1.3 History of Analog IC Design Automation	3
1.3.1 CAD Tools for Analog-Mixed Signal Design	3
1.3.2 Design Automation Methodologies	3
1.3.2.1 Topology Selection Approaches	3
1.3.2.2 Circuit Sizing/Optimization Approaches	4
1.3.2.3 Generator-Based Approaches	6
1.4 Our Work	6
1.4.1 Implementation Constraints	6
1.4.2 Thesis Outline	7
2 Algorithm Overview	8
2.1 Definition of the Problem	8
2.2 Population-Based Methods: Benefits and Drawbacks	9
2.3 Model for Imitating the Oracle	11
2.4 Algorithm	12
3 Experiments	15
3.1 Schematic Design of a Two Stage Amplifier	15
3.2 Layout Design of a Two Stage Amplifier and Comparison against an Expert Design	19
3.3 End-to-End Layout Optimization of an Optical Receiver Link	21
4 Conclusion and Future Work	27

A Framework Implementation	28
A.1 Overview of Framework Structure	28
A.2 Evaluation Engine	29
A.3 Model	29
A.4 Black-box Optimization	30
A.5 Decision Box	31
Bibliography	32

List of Figures

2.1	Illustration of improvement of Oracle against EA. Each iteration is equivalent to adding some number of new designs to the population. The region between the two curves represents room for improvement.	10
2.2	DNN's model used in the system, $\theta = [\theta_f, \theta_1, \dots, \theta_l]$ contains the parameters of the DNN. $\mathcal{M}_\theta(D_A, D_B)$ is the output probabilities of the DNN parametrized by θ for inputs D_A and D_B . $\mathcal{M}_\theta(D_A, D_B; i)$ denotes the predicted probability for the i^{th} specification. Note that \succeq denotes preference, not greater than.	11
2.3	High level architecture of our optimizer	13
3.1	Schematic of a vanilla two stage op-amp	16
3.2	a) Average cost of top 20 individuals across number of iterations. Each iteration corresponds to adding 5 designs to the population. b) Average cost of top 20 individuals across number of simulations.	18
3.3	Two stage op-amp with negative g_m load	20
3.4	Convergence curve of the two stage op-amp done in BAG. Orange shows the minimum cost across iteration step and blue shows the average of cost in top 20 individuals.	20
3.5	Optical receiver schematic	22
3.6	a) Overdrive test recovery simulation curves for DTSA b) Probability of outputting a one vs. V_{in} . We can use the cumulative density function of a Gaussian to estimate the standard deviation of the noise.	23
3.7	a) Input and output signals for measuring the eye height and thickness, the input is a small signal pulse with an amplitude of target input sensitivity, and with a width of T_{bit} for the target data rate. The output curve is sampled at time instances shown relative to the main cursor (maximum of output) b) Equations used for estimating eye height and thickness to express the fidelity of eye diagram	24
3.8	Sample layouts of the optical link receiver circuit with different cost values of 0, 0.1, and 0.2 for A, B, and C, respectively.	25
A.1	The overview of framework modules	28

List of Tables

3.1	Objective of design and performance of solutions found using different approaches for the two stage op-amp example	16
3.2	Summary of number of operations involved in the process of each approach	17
3.3	Performance of expert design methodology and our approach	20
3.4	Design Performance for optical receiver design for $C_{PD} = 20fF$, $I_{min} = 3\mu A$, Data Rate = $10Gbit/s$	26

Acknowledgments

I've been fortunate enough to come to UC Berkeley for my graduate studies and become part of this open community to new ideas and leverage the freedom of research it provided. In that regard, I would like to emphasise my special thanks to my advisor, Vladimir Stojanovic' who has made this project possible with his constant advice and support. I also would like to thank Pieter Abbeel whose guidance has been extremely valuable through out this research project. I hope we can conduct many more exciting projects together through out my continuing journey in PhD studies.

Additional thanks to all my peers in team Vlada for their inputs and feed-backs. Thank you Nick for providing such an amazing collection of generators. Thanks to Acacia team, Krishna, Sidney, Roucheng, Zhaokai, Eric Jan who made my first tape-out experience possible. Especial thanks to Eric Chang et al. for creating BAG which, in my mind, had the most impact on significance of this research.

Last but not least, thanks to my dearest wife, Yasi, whose support has been unimaginable through our ups and downs, for keeping me sane in all years we have known each other, for making sure I don't starve to death while doing research (I really can!), and most of all for being my best friend in all circumstances.

Chapter 1

Introduction¹

1.1 Integrated Circuit Market Overview

The Microelectronic industry has been significantly evolved over the past years, driven by an ever-increasing demand in machine learning (ML) and internet of things (IoT) applications, requiring many functionalities to be integrated on a single System-on-Chip (SoC) solution. Today's generation of SoCs integrate massive number of digital computational cores (i.e. CPU, GPU, and neural processing units) with an increasing number of analog-mixed signal (AMS) and radio frequency (RF) components (i.e. radio transceivers, sensors, power regulators, high speed IOs etc.) on the same chip. This makes the design and verification process exponentially more convoluted than before and also prone to more errors.

Although the core functionality is typically driven by the digital cores, the critical path for errors and performance tends to be in the analog part. This is due in part to the differences in the maturity levels of CAD tools in digital and analog design. Well-established practices supported by well-defined automated synthesis methodologies and tools exist for digital design. However analog design, due to the higher sensitivity of design to parameters and the fabrication process, requires extensive, skilled manual labor. It is not as modular as digital design, and as a result, a standard cell library of sub-blocks cannot be used.

1.2 Analog IC Design Flow

As was mentioned in the previous section, AMS design requires more manual labor than digital design. In the digital world, signals are more tolerant to noise and the information is carried around in discrete levels of voltage or current with a large tolerance range. The behavior of such circuits is defined by Boolean algebra, and the timing information is also modular and defined for a library of standard cells. The optimization problems are well-defined and therefore automation tools are much easier to be developed.

¹Most part of this chapter, especially the information on the background on design automation history is adapted from [3](#)

In AMS circuits however, information is conveyed through a continuum of (typically small) values, and therefore second and third order effects which were ignored in digital design becomes a problem in analog; issues like, non-linearity, devices going into their non-linear region, matching between devices, effects of noise, etc. have to be considered.

To cope with these issues designers have to adopt top-down design and bottom-up verification methodologies. This means that they start the design flow by top level behavioral modeling and hierarchically move down and verify the performance and behavior at each step all the way back up with different levels of abstraction. To do so, they start with system level abstraction, verification, and architecture exploration until they settle on the AMS sub-blocks and their interaction with each other. The agile IC development flow requires designers to then use dummy or semi-realistic sub-blocks as the place holder for the circuits that will be actually designed later and verify the entire system-level flow with the behavioral models that they expect from each sub-block. This way they can catch system-level problems early on as the sub-blocks evolve over the design procedure. This processes is then repeated for lower level sub-blocks (which have their own sub-blocks) until they reach the actual device and layout level design.

As designers move down the hierarchy, they have to re-verify their assumptions on the higher levels and if something does not behave properly re-design the lower levels. The number of iterations in verification and design tends to increase as designers get to layout. This is largely in part due to the domination of layout parasitics in determination of circuit performance. Therefore, Designing the entire system involves many iterations with human experts exploring the complex multi-dimensional design space. Besides the large number of iterations, simulation will also get slow as designer moves down from behavioral simulations to device and layout level.

1.3 History of Analog IC Design Automation

Over the ages of semiconductor industry, researches have tried an assortment of approaches to automate AMS design problem in different stages. However, CAD tools have not been following the semiconductor technology at the same pace, resulting in increasing price of developing state-of-the art SoCs. In this section, we will briefly discuss available CAD tools and some automation procedures proposed over the years.

1.3.1 CAD Tools for Analog-Mixed Signal Design

An improved but yet limited degree of automation is supplied by the use of a CAD methodology which involves the integration of one or more mature CAD tools into a flow. One of the most known CAD tools is the Cadence Virtuoso platform which is composed by a set of integrated circuit tools that cover all the stages, from the schematic to the layout. Apart from the Composer schematic editor, Cadence Virtuoso includes a high accuracy circuit simulator, like Virtuoso Spectre that is usually used at the device level, a layout editor and layout verification tools that implement the three different phases of layout process, the design rule checking (DRC), layout versus schematic (LVS) and parasitic extraction (RCX). Additionally, the system level analog behavioral descriptions may be simulated with the System-Verilog simulator. These design management platforms are a valuable help in analog integrated circuit design but they are still far behind the development stage of design automation tools already available for digital design. [3]

1.3.2 Design Automation Methodologies

Automating analog IC design is not a novel idea by itself and has been around for over twenty years. An excellent survey about these efforts has been done by [22]. We can divide the previous approaches in the following category and sub categories:

1.3.2.1 Topology Selection Approaches

In these flavor of approaches the goal is to basically induce the topology based on some criteria either from a library or by composing sub-components together and forming more complex ones. To name a few of these efforts, in IDAC [6] the decision is taken directly by the designer. Heuristic rules have been used in the first attempts by [10]-[13], to automate the topology selection task. The tool [23] uses fuzzylogic based reasoning to select one topology among a fixed set of alternatives. The decision rules are introduced by an expert designer or automatically generated by means of a learning process. Another method comprises computing the feasible performance space for each topology within the library and, then compare with the desired performance specs [8]. A different method consists of combining the topology selection with the device sizing task and employing an optimization based approach by [15] using genetic algorithms.

As was mentioned earlier, these approaches have several problems. New generation of designs are far more complicated to be supported by a predefined set of libraries. Most of these approaches demonstrated their usefulness in design of operational amplifiers which is not as complicated as today's practical circuits. On top of that, changing the technology requires new rounds of library generation, which might be time consuming. Additionally, all of these methods do not consider layout effects which drastically change the performance of the circuits.

1.3.2.2 Circuit Sizing/Optimization Approaches

These approaches typically cast the the sizing problem as an optimization counterpart and seek to satisfy the objective. In many cases, the optimization formulations do not consider all practical and non-ideal phenomena in circuits, leaving the burden on analog designers to set the constraints properly, and guide the optimization procedure. Therefore, it is easier for analog designers to just directly solve the optimization themselves rather than codifying all overt design criteria (i.e. matching of layout, transistors not being in linear region, etc.), as well as performance metrics (i.e. gain, bandwidth, etc.).

A lot of these readily assumed design metrics are often hard to codify, either because they cannot be captured by plain simulations (i.e. matching) or they need a specialized expensive simulation (i.e. Monte Carlo simulations for matching). Some of these "hard-to-codify" metrics come from layout methodologies that designers use because they are "safe" and proven to be functional (i.e. how to use common centroid layout procedure for current mirrors to ensure matching). Beside these, designers tend to realize missing some constraints till the optimization framework exploits those under-defined constraints which, typically, takes a long time. [3], [22] divides the circuit sizing approaches into the following categories:

Equation-Based Methods

These approaches model the circuit behavior as analytical formulations and the objective is to solve an equation-based optimization problem using numerical methods.

Some of the most relevant approaches are [13], [11], [18], [9], [12]. This approach presents the advantage of allowing a performance evaluation speed-up (short evaluation time). The main drawback is that analytical models have to be used to derive the design equations for each new topology and, despite recent advances in symbolic circuit analysis [24], [24], not all design characteristics can be easily captured by analytic equations. The approximations introduced in the analytic equations yields low accuracy designs especially in complex circuits designs. These methods are obsolete and cannot guarantee accuracy of the solution.

Simulation-Based Methods

Black box optimization methods like [17], [23], [19], [14] and [15] consist of using a simulation core in the inner loop for circuit evaluation. This gives us a generic framework,

independent of circuit topology (compared to equation-based methods), for high accuracy evaluation of circuits.

Despite the benefits, these methods are computationally expensive and time-consuming to run, especially for larger, more complex circuits. Nevertheless, a key difficulty is that the analog design problem, with all the involved design knowledge and heuristics, has to be formulated as an optimization problem, which often presents a high threshold for using a circuit-sizing tool. Therefore, sample efficiency, and short convergence time is a determining factor in these approaches. On the other hand, all the previous work in this area has been targeted for schematic-level simulations, and layout parasitic effects have been considered only outside of the optimization loop.

Machine Learning-Based Methods

To improve the convergence of these methods, many of the previous work such as [2], [1], and [25] has tried to model the behavior of the circuit to be optimized, by a machine learning mechanism to quickly evaluate the performance for a specific set of design parameters, essentially replacing the simulator's long evaluation cycles.

In [1] a learning tool based on support vectors machines (SVM) is used to represent the performance space of analog circuits. Based on the knowledge acquired from a training set, the performance space is modeled as mathematical relations translating the analog functionality. SVMs are trained with simulation data, and false positives are controlled based on a randomized testing procedure.

[25] presents a performance macro-model for use in the synthesis of analog circuits based in a neural network approach. On the basis of this mathematical model is a neural network model approach that, once constructed, may be used as substitute for full SPICE simulation, in order to obtain an efficient computation of performance parameter estimates. The training and validation data set is constructed with discrete points sampling over the design space. The work explores several sampling methodologies to adaptively improve model quality and applies a sizing rule methodology in order to reduce the design space and ensure the correct operation of analog circuits.

The issue with almost all of these methods is that they require a lot of sampling points in their training data to build a model. Nevertheless, good design points tend to be a small subset of the extremely large multi-dimensional space. Hence, despite the high overall accuracy of the model, the precision in the regions of good designs can be low, resulting in inaccurate models in regions that designers actually care about. Also, similar to previous methods, these approaches have mainly modeled schematic behavior of circuits. Training considering layout effects, can become infeasible using these approaches because of the long simulation time of each circuit instance.

1.3.2.3 Generator-Based Approaches

In these approaches, instead of optimizing circuit sizing objectives, the designer codifies the step by step sizing and design process they pursue in a parametrized format called "generators". In other words, the input to a "layout generator" script is the parameters of devices (i.e. number of unit widths, number of fingers, etc.) and the output is the associated layout in a particular technology. Following this strategy we can capture all layout constraints (i.e. matching, sharing drain/sources to reduce parasitics etc.) in a programmable, reusable fashion and port a layout to new technologies with just a push button.

Berkeley Analog Generator (BAG) [5] is a framework for development of process-portable AMS circuit generators. It has multiple layout engines that provides designers with the necessary API to capture their layout procedure for analog as well as custom digital circuits. It also provides a coherent environment for design verification in a Python-based language. Designer can develop "schematic generators" which essentially generates the schematic of a design in a parametrized way. They can then run LVS and DRC directly from the same Python environment and also run test benches associated with verification of the blocks. They can also leverage objected oriented programming (OOP) features of Python to extend and reuse other designers' codes. In summary it provides a convenient interface for designers to specify their layout, design, and verification methodologies.

1.4 Our Work

1.4.1 Implementation Constraints

In this work our objective is to build an optimization framework that directly optimizes layout. According to the previous sections of this chapter, we find that the following items should be the key features of such a framework:

1. This framework should be based on reliable simulation tools to be accepted by the community of circuit designers. Because equations are not generalizable across circuits and technologies, and have limited accuracy. Moreover, simulation tools are mature and are the mainstream tool for verification among designers.
2. Post layout simulation is expensive in terms of simulation time. Therefore, the optimization framework needs to be sample efficient Otherwise the tool will be in contrast to boosting the productivity of designers.
3. From designer's perspective, it should be convenient to codify design criteria. It should not create more hassle for designers trying to optimize AMS circuits.
4. It is not necessary to give the best design possible. In many cases having some design that just satisfies the constraints can be extremely useful in architecture exploration.

1.4.2 Thesis Outline

In this work, we will extend BAG's API to incorporate this optimization framework for design purposes. The vision is that once designers have codified their layout and verification methodology, this tool will explore the design space in a time efficient manner (probably overnight when designers are asleep) and will candidate design solutions that follow the layout and verification constraints they defined.

In chapter [2](#) we will discuss the theory of our algorithm and how leveraging a Deep Neural Network (DNN) can help us boost the sample efficiency of conventional stochastic optimization algorithm (i.e. genetic algorithm, simulated annealing, etc.). In chapter [3](#) we will demonstrate the effectiveness of this approach on several example AMS circuits, including a vanilla version of an optical link optimized from high-level specifications, and in chapter [4](#) we will conclude the work and propose future directions. Appendix [A](#) will be discussing, in more details, the implementation of the algorithm using available Python packages.

Chapter 2

Algorithm Overview

2.1 Definition of the Problem

The objective in analog circuit design is usually to minimize one figure of merit (FOM) subject to some hard constraints (strict inequalities). For instance, in op-amp design the objective can be minimizing power subject to gain and bandwidth constraints. However, in practice there is also a budget for metrics in the FOM (i.e power less than 1mW). Therefore, the optimization can be rephrased as a constraint satisfaction problem (CSP) where the variables are circuit's geometric parameters, and outputs are specifications of the circuit topology. Designers can always tighten the budgets to see if there is any other answer with a better FOM that meets their needs.

However, in cases where there is no feasible solution to a CSP, designers still prefer to know which solutions are nearly satisfactory to gain insight into which constraint can be adjusted to satisfy their needs.

With this in mind we define the following non-negative cost function where finding the zeros is equivalent to finding answers to the CSP problem. If no answer exists, the minimum of this cost and the non-zero terms can give insight about which metrics are the limiting factors:

$$cost(x) = \sum_i w_i p_i(x) \quad (2.1)$$

where x represents the geometric parameters in the circuit topology and $p_i(x) = \frac{|c_i - c_i^*|}{c_i + c_i^*}$ (normalized spec error) for designs that do not satisfy constraint c_i^* , and zero if they do. c_i denotes the value of constraint i at input x , and is evaluated using a simulation framework. c_i^* denotes the optimal value. Intuitively this cost function is only accounting for the normalized error from the unsatisfied constraints, and w_i is the tuning factor, determined by the designer, which controls prioritizing one metric over another if the design is infeasible.

2.2 Population-Based Methods: Benefits and Drawbacks

Population based methods have been extensively studied in the past in the application of analog circuit design automation. These methods usually start from an initial population and iteratively derive a new population from the old one using some evolutionary operations (i.e. combination, mutation). Some selection mechanism then picks the elites of the old and new populations for the next generation, a process known as elitism. This process continues until the average cost of current population reaches a minimum.

While this could work in principle, it is very sample inefficient, and prone to instability in convergence. Therefore, the process must be repeated numerous times due to its stochastic nature. As a result, these methods are not suitable for layout based optimizations where simulation takes a significant amount of time.

The sample inefficiency arises from two factors. Firstly, the majority of the new population will only slightly improve on their ancestors, and as the population improves, the difficulty of replacing old designs increases. Therefore, it would take many iterations until the children evolve enough to surpass the average of the parents. Much of the previous work seeks to improve this by focusing on modifying the evolutionary operations such that they would increase the probability of producing better children, while preserving the diversity [20]. Unfortunately, these methods have not been able to sufficiently improve the sample efficiency to accommodate the post-layout simulations. Secondly, many of these methods only look at the total cost value and do not consider sensitivity of the cost to each design constraint, meaning that they do not account for how each specification metric is affecting the overall cost. Expert analog designers usually do this naturally by prioritizing their design objectives depending on what constraint most limits their design. Considering only the total cost value can be misleading and may obfuscate useful information about the priority of optimizing the metrics.

To address the first issue, if we had access to an oracle which could hypothetically tell us how two designs were compared in terms of each design constraint, we could use it to direct the selection of new designs. Each time a new design is generated we can run the oracle to see how the new design compares to some average design from the previous generation. In this paper we devise a DNN model that can imitate the behaviour of such an oracle.

To address the second problem, we can look at the current population and come up with a set of critical specifications (i.e specifications that are the most limiting and should be prioritized first). Each step that we query the oracle, we only add designs that have better performance than the reference design in all metrics in the critical specification set. The important point to note is that once a metric enters the critical specification set it never becomes uncritical, as we do not want to forget which specifications derived the selection of population before the current time step. For finding the critical specification at each time step we use a heuristic which is best described by the pseudo code in algorithm 1.

We can realize the oracle by a simulator and by using the aforementioned heuristics, we

Algorithm 1 Pseudo-code of the heuristic used for updating critical specification list

Given population \mathcal{B} , specification list \mathcal{S} , critical specification list \mathcal{CS} (empty at first), a reference index k (i.e 10)

if $\mathcal{CS}.\text{empty}()$ **then**

$\tilde{\mathcal{B}} \leftarrow \text{sort } \mathcal{B} \text{ by } \text{cost}(x) = \sum_{i \in \mathcal{S}} w_i * p_i(x)$

else

$\tilde{\mathcal{B}} \leftarrow \text{sort } \mathcal{B} \text{ by } \text{cost}(x) = \sum_{i \in \mathcal{CS}} w_i * p_i(x)$

end if

critical_spec $\leftarrow \arg \max_{i \in \mathcal{S}} \max_{x \in \tilde{\mathcal{B}}[0:k-1]} p_i(x)$

$\mathcal{CS}.\text{append}(\text{critical_spec})$

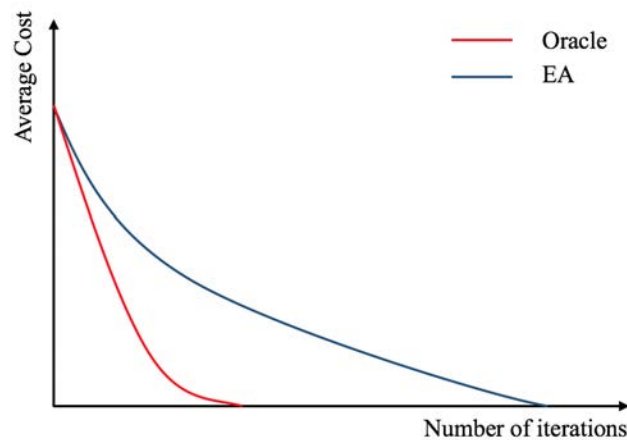


Figure 2.1: Illustration of improvement of Oracle against EA. Each iteration is equivalent to adding some number of new designs to the population. The region between the two curves represents room for improvement.

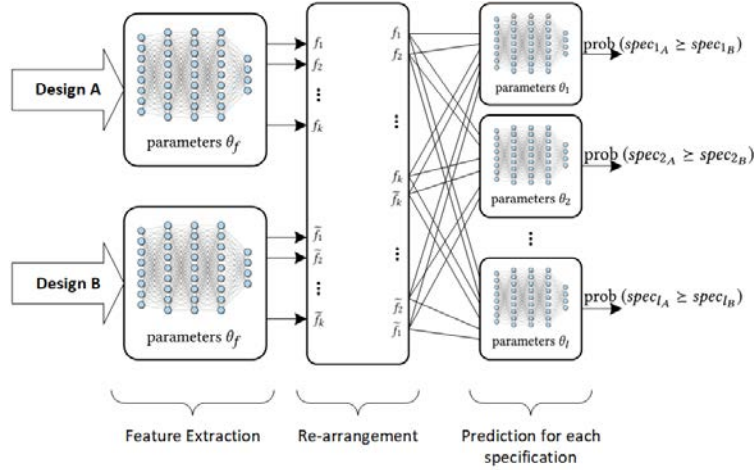


Figure 2.2: DNN’s model used in the system, $\theta = [\theta_f, \theta_1, \dots, \theta_l]$ contains the parameters of the DNN. $\mathcal{M}_\theta(D_A, D_B)$ is the output probabilities of the DNN parametrized by θ for inputs D_A and D_B . $\mathcal{M}_\theta(D_A, D_B; i)$ denotes the predicted probability for the i^{th} specification. Note that \succeq denotes preference, not greater than.

can decide whether to add a new design to the population or not. Figure 2.1 illustrates the notional convergence performance of this oracle compared to the same evolutionary algorithm without this modification. This illustration shows that the oracle can significantly reduce the number of iterations for convergence if we knew what designs to add and what designs to reject.

However, we cannot use this oracle if we want to scale our method to do layout-level optimizations on more complex circuit topologies with larger design spaces and more expensive simulation runs. This is because the oracle has to run simulations for all generated instances to determine which designs to add or reject and therefore, there is no real benefit in the number of simulations that it runs. In the next section we propose a DNN structure that can imitate the behavior of the oracle while reducing the required number of simulation runs.

2.3 Model for Imitating the Oracle

For imitating the oracle there are multiple options. First, we can have a regression model to predict the cost value and then use this predicted value to determine whether or not to accept a design. The cost function that the network tries to approximate can be an extremely non-convex and ill-conditioned. Thus, from a limited number of samples it is very unlikely that it would generalize well to unseen data. Moreover, the cost function captures too much information from a single scalar number, so it would be hard to train.

Another option is to predict the value of each metric (i.e. gain, bandwidth, etc.). While the individual metric behaviour can be smoother than the cost function, predicting the actual

metric value is unnecessary, since we are simply attempting to predict whether a new design is superior to some other design. Therefore, instead of predicting metric values exactly, the model can take two designs and predict only which design performs better in each individual metric. Figure 2.2 illustrates the model architecture used for imitating the oracle.

The model consists of a feature extraction component comprised of only fully connected layers which are the same for both Design A and Design B. For each specification there is a sub-DNN that predicts the preference over specifications using fully connected layers. There is a subtle constraint that the network should predict complementary probabilities for inputs $[D_A, D_B]$ vs. $[D_B, D_A]$ (i.e. $\mathcal{M}_\theta(D_A, D_B) = 1 - \mathcal{M}_\theta(D_B, D_A)$) meaning that there should be no contradiction in the predicted probabilities depending on the order by which the inputs were fed in. Therefore to ensure this property holds and to make the training easier, we can impose this constraint on the weight and bias matrices in the decision networks. To do so, each sub-DNN's layer should have even number of hidden units, and the corresponding weight and bias matrices should be symmetric according to the following equations.

$$y_{m \times 1} = \mathbf{W}_{m \times 2k} x_{2k \times 1} + b_{m \times 1}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{\frac{m}{2} \times 2k} \\ \widetilde{\mathbf{W}}_{\frac{m}{2} \times 2k} \end{bmatrix} \begin{bmatrix} x_{k \times 1} \\ \widetilde{x}_{k \times 1} \end{bmatrix} + \begin{bmatrix} b_{\frac{m}{2} \times 1} \\ \widetilde{b}_{\frac{m}{2} \times 1} \end{bmatrix}$$

Where we have,

$$\begin{aligned} \widetilde{\mathbf{W}}(i, j) &= \mathbf{W}\left(\frac{m}{2} - 1 - i, 2k - 1 - j\right) \\ \text{for } i &= 0, \dots, \frac{m}{2} - 1 \text{ and } j = 0, \dots, 2k - 1 \\ \widetilde{b}(i) &= b\left(\frac{m}{2} - 1 - i\right) \text{ for } i = 0, \dots, \frac{m}{2} - 1 \end{aligned}$$

If the weight and bias parameters are set as above, when the input order is changed from $[D_A, D_B]$ to $[D_B, D_A]$ the very first x vector is changed from $[f_1, \dots, f_k, \widetilde{f}_k, \dots, \widetilde{f}_1]$ to $[\widetilde{f}_1, \dots, \widetilde{f}_k, f_k, \dots, f_1]$. Thus, for the last layer that has two outputs, the sigmoid function will produce $1 - \mathcal{M}_\theta(D_A, D_B)$ instead of $\mathcal{M}_\theta(D_A, D_B)$.

To train the network, we construct all permutations from the buffer of previously simulated designs and evaluate their performance in each metric. We then update network parameters with stochastic gradient descent.

2.4 Algorithm

To put everything together, Figure 2.3 illustrates the high level architecture of the optimizer system. We use the current population and perform some specific evolutionary operations to get the next generation of the population; but we do not simply simulate them

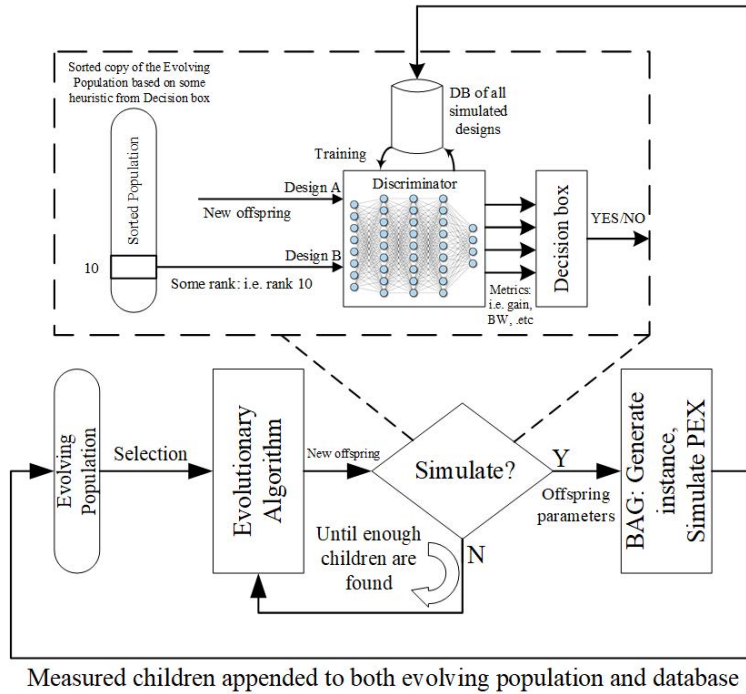


Figure 2.3: High level architecture of our optimizer

and consider them as the next generation. We use the DNN to predict if they will be better compared to some reference design already within our current population, and if the answer is positive we simulate the designs, call them the next generation, and proceed with the evolutionary algorithm.

The children may have a distribution mismatch from the data that DNN was trained on. To mitigate the distribution drift, each time we add new children we evaluate them with the actual simulator and re-train the model with correct labels so that the buffer is updated with correct labels for next steps of training. This idea is very similar to DAgger [21] except that we do not relabel all of the children, rather we only relabel the accepted ones. Algorithm 2 shows the entire algorithm, step by step.

Algorithm 2 Pseudo-code for the entire algorithm

Given Some evolutionary operations \mathcal{E} ▷ i.e. CEM 4
 Given Some Initial buffer of random simulated designs \mathcal{B}
 Given reference index k ▷ i.e. 10
 Given DNN \mathcal{M}_θ parametrized by θ
 update θ ▷ i.e. 10 epochs
while num_iter \leq max_num_iter **do**
 Get critical specification list \mathcal{CS} according to the heuristic
 $\tilde{\mathcal{B}} \leftarrow$ sort \mathcal{B} by $cost(x) = \sum_{i \in \mathcal{CS}} w_i * p_i(x)$
 $\mathcal{D}_{ref} = \tilde{\mathcal{B}}[k]$
 list of new children $\mathcal{L} = []$
 while $\mathcal{L}.length < 5$ **do** ▷ i.e. until 5 children are approved
 $\mathcal{D}_{new} \leftarrow \mathcal{E}.generate(\mathcal{B})$ ▷ generate a new design
 $\mathcal{P} \leftarrow M_\theta(\mathcal{D}_{new}, \mathcal{D}_{ref})$
 if $\mathcal{P}[i] = 1, \forall i \in \mathcal{CS}$ **then**
 Run simulation on \mathcal{D}_{new}
 $\mathcal{L}.append(\mathcal{D}_{new})$
 else
 Continue
 end if
 end while
 $\mathcal{B} \leftarrow \mathcal{E}.select(\mathcal{B} + \mathcal{L})$
 update θ ▷ i.e. 10 epochs
end while

Chapter 3

Experiments

In this section we study a variety of experiments which clarifies some aspects of the algorithm and illustrates its capabilities on a variety of circuits.

3.1 Schematic Design of a Two Stage Amplifier

First, to clarify the convergence behavior and benefits of the algorithm, we use a simple two stage op-amp evaluated only through schematic simulation using 45nm BSIM models on NGSPICE.

The circuit's schematic is shown in Figure 3.1. The objective is to find the size of transistors and the value of the compensation capacitor such that the described circuit satisfies the requirements set in table 3.1. We fixed the length and width of the unit sized transistors to 45 nm and 0.5 μm , respectively, and for size of each transistor we limit the number of fingers to any integer number between 1 to 100. For compensation we also let the algorithm choose C_c from any number between 0.1pF to 10pF with steps of 0.1fF. The grid size of the search space is 10^{14} . A given instance is evaluated through DC, AC, CMRR, PSRR, and transient simulations, which in total takes one second for each design. Therefore, brute-force sweeping is not practical even in this simple example. However, This short simulation time allows us to do comparisons against the vanilla genetic algorithm and the oracle. Note that these methods are not feasible for layout-based simulations, however, since each takes multiple minutes per design to evaluate.

Table 3.1 shows the performance of the minimum cost solution found by different approaches. In this example, all approaches found a solution satisfying all specifications, but this is not guaranteed as it depends on the feasibility of the specifications and also the stochastic behaviour of the evolutionary algorithms. We can always adjust exploration vs. exploitation of the evolutionary algorithms by the mutation rate, but this will increase convergence time of all of the approaches. However, if training parameters are set properly in our approach we will not need as many simulations as the oracle or the vanilla evolutionary algorithms.

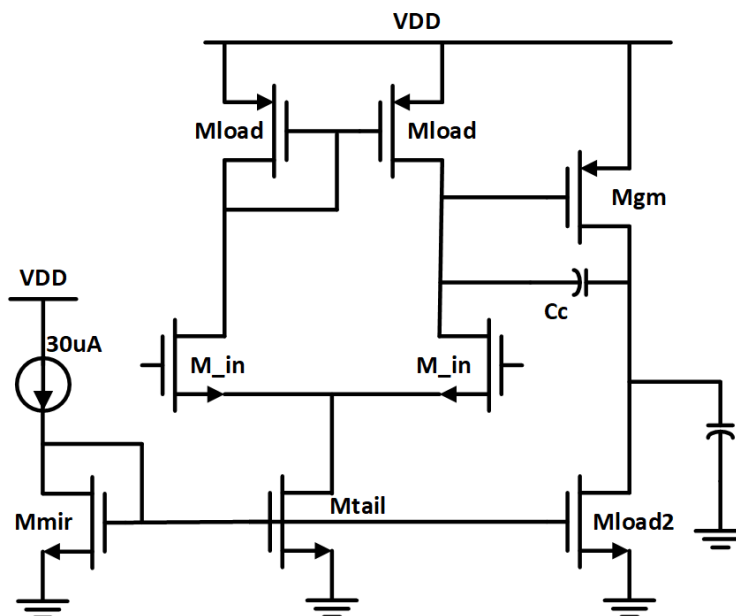


Figure 3.1: Schematic of a vanilla two stage op-amp

Table 3.1: Objective of design and performance of solutions found using different approaches for the two stage op-amp example

	Requirement	Evolutionary	Oracle	Ours
Gain	>300	323	314	335
f_{unity} [MHz]	>10	10.83	10.66	10.2
Phase Margin [°]	>60	60.7	60.83	62
$t_{settling}$ [ns]	<90	59.9	83.5	62
CMRR [dB]	>50	53	54	54
PSRR [dB]	>50	57	56	57
Systematic Offset [mV]	<1	0.823	0.94	0.32
Ibias [μA]	<200	188	158	148

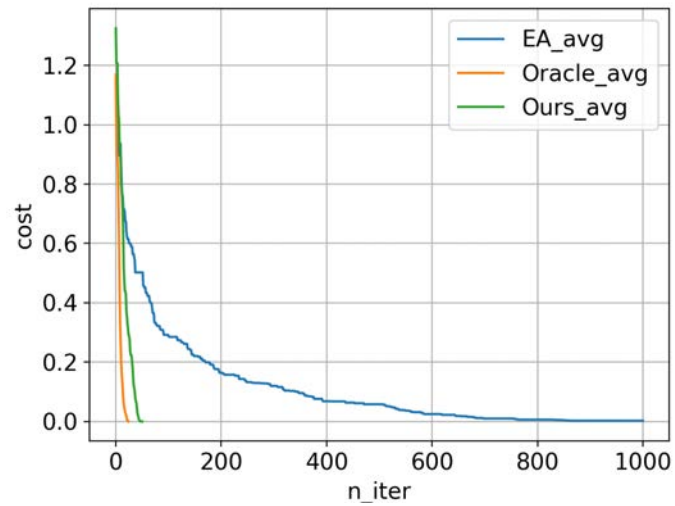
Table 3.2: Summary of number of operations involved in the process of each approach

	# of NN Queries	# of Re-training	# of Simulations
Simple Evolution	-	-	5424
Oracle	-	-	3474
Ours	55102	50	241

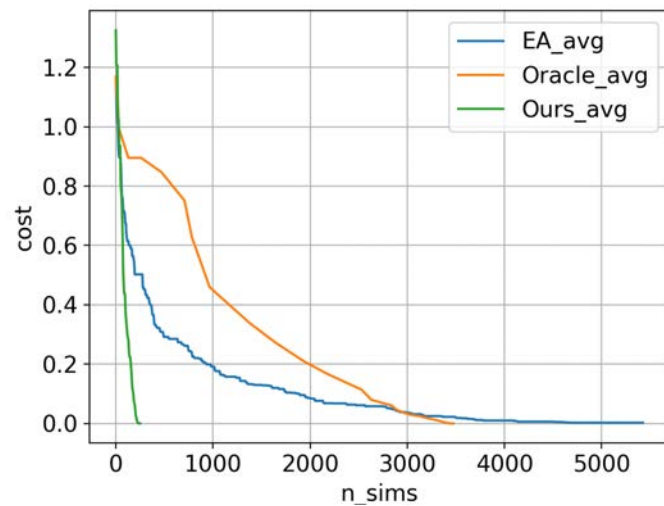
To avoid over-fitting and being certain about false positive and negatives, we can leverage Bayesian DNNs within our model [16], which can estimate the uncertainty regarding the decisions. In this specific example we use drop out layers which can be considered as Bayesian DNNs with Bernouli distributions [7]. During inference we sample the model 5 times and average the probabilities to reduce the uncertainty about the decisions.

Figure 3.2a shows the average cost of the top 20 designs (parameter k is 20 in algorithm 2) in the population for the oracle, vanilla evolutionary and our algorithm. Each time only 5 designs are added and the evolutionary operations are the same for all experiments. We ran our approach on multiple random initial seeds to ensure robustness in training and performance. We note that in terms of number of iterations, the oracle is the fastest, but in fact the oracle runs simulation on all generated designs to make selection decisions. Therefore, it is impractical to run it on post-layout simulation on more complex circuits. Figure 3.2b shows performance in terms of number of simulations. We see that our approach is efficient by at least a factor of 10 in this simple example.

Table 3.2 shows a summary of number of operations in our simple example. We note that our approach can cut down a lot of impractical simulations at the cost of more time spent on training and inference of a DNN. With recent advancements in hardware for machine learning and use of GPUs the time spent on training and inference can significantly be reduced. When we want to scale up to more intricate circuits there are two factors that makes our approach advantageous. Firstly, when we do layout optimization, simulation drastically increases proportionally to the circuit size. Also, more complicated circuits have larger design space and it will become even more critical to prune out impractical regions of design space as we scale up. Therefore, in terms of scaling to layout optimization our approach seems to be promising.



(a)



(b)

Figure 3.2: a) Average cost of top 20 individuals across number of iterations. Each iteration corresponds to adding 5 designs to the population. b) Average cost of top 20 individuals across number of simulations.

3.2 Layout Design of a Two Stage Amplifier and Comparison against an Expert Design

This example is presented to compare an expert-designed circuit with our algorithm’s design. The op-amp’s topology, and the cardinality of search space is shown in Figure 3.3, with each array denoting how many parameters were considered for design. For example for Mref, 20 values of n_fingers were considered. In total, this design example has an 11 dimensional exploration space with size of 3×10^{13} .

The topology is a standard Miller compensated op-amp, in which the first stage contains diode-connected and negative-gm loads. The design procedure is more cumbersome than the previous two stage example, mainly because of the positive feed back. A scripted design procedure for this topology is included as part of BAG to exemplify codifying expert driven design methodologies. The design script is able to find the proper transistor sizing while considering layout parasitic effects using a closed loop design methodology. The inputs to the design script are specifications of phase margin and bandwidth, and the objective is to maximize gain. In this circuit the resistor and capacitor are schematic parameters, while all transistors and all connecting wires use the GF14 nm PDK extraction model.

Table 3.3 shows a performance summary of our approach compared to that of the design generated by the design script. The script is unable to meet the gain requirement due to a designer-imposed constraint that the negative g_m should not cancel more than 70% of the total positive resistance at first stage’s output. This constraint arises from a practical assumption that there will be mismatch between the negative g_m ’s resistance and the overall positive resistance, due to PVT variations. Thus, the circuit can become unintentionally unstable, and therefore during design we leave some margin to accommodate these prospective random variations. We have the option of imposing a similar constraint to equate the design spaces, or we can run simulations over process and temperature variations to ensure that our practical constraints are not too pessimistic.

For our approach, the initial random population size is 100 with a best cost of 0.3. We ran every simulation on different PVT variations, and recorded the worst metric as the overall performance value. Figure 3.4 illustrates how the average cost in top 20 designs changes over time. Reaching a solution with our approach took 3 hours including initial population characterization, whereas developing the design script takes 4-7 days according to the expert. The DNN was queried 3117 times in total (equivalent to 6 minutes of run time on our compute servers) and we only ran 120 new simulations in addition to the initial population of size 100 (each of which takes on average 48 seconds to run). Moreover, the complexity of developing a design script forces the designer to limit the search space in an effort to make the process feasible and a generic design algorithm that properly imposes high-level specifications onto a large system is immensely difficult to generate. We will see an example of such systems and our approach’s solution in the next section.

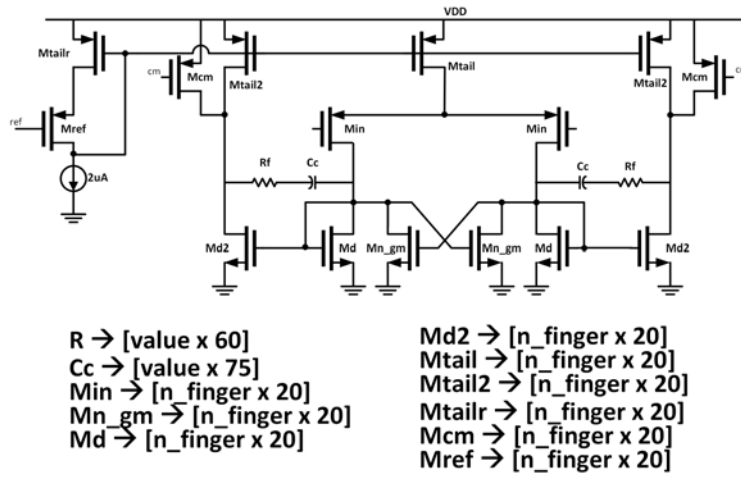


Figure 3.3: Two stage op-amp with negative g_m load

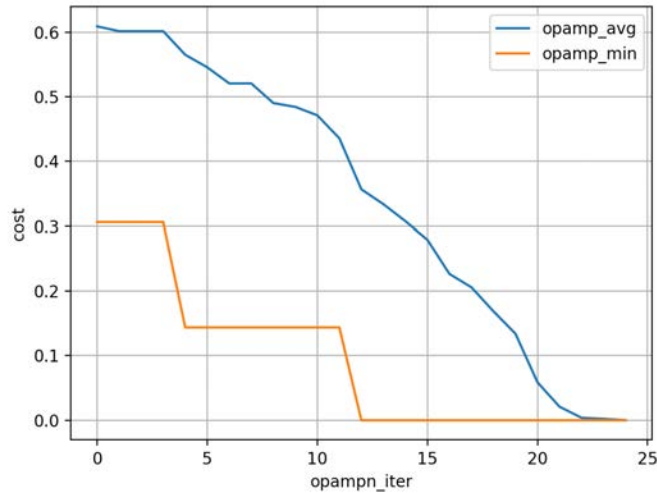


Figure 3.4: Convergence curve of the two stage op-amp done in BAG. Orange shows the minimum cost across iteration step and blue shows the average of cost in top 20 individuals

Table 3.3: Performance of expert design methodology and our approach

	Requirement	Expert	Ours
f_{unity}	>100 MHz	382 MHz	159 MHz
pm	> 60°	64°	75°
gain	>100 (for ours)	42	105

3.3 End-to-End Layout Optimization of an Optical Receiver Link

The following experiment highlights the capabilities of our approach in handling complex analog/mixed signal design problems using post-layout simulations. We demonstrate a differential optical link receiver front-end with a one tap double tail sense amplifier (DTSA) in the end. The circuit is shown in figure 3.5 with design space parameters at the bottom. The goal is to design this circuit from very high level specifications, namely, data rate, power consumption, and minimum sensitivity for a given bit error rate (BER).

Automating characterization of instances of this circuit is the key in setting up the environment prior to running the algorithm. The following steps are crucial to get performance metrics on each design. First we instantiate the DTSA's layout, schematic, and extracted netlist. We then run overdrive test recovery to characterize the transient behaviour of DTSA. Figure 3.6a shows a typical overdrive test recovery curve for a given comparator. We specifically measure v_{charge} , v_{reset} , and v_{out} in the time instances relative to the edge of the clock as shown in the figure. By specifying these three numbers as well as a minimum v_{in} (i.e 1 mV), we can describe the performance of the comparator at a given data rate. To get the noise behaviour of the comparator, we run numerous transient noise simulations for several cycles while sweeping input voltage from a small negative voltage to a small positive voltage. We can then fit a normal Gaussian distribution to the estimated probability of ones in each transient run and get an estimation of the input referred voltage noise of the DTSA. Figure 3.6b illustrates this simulation procedure. We then take the entire system's extracted netlist and characterize the behaviour of the analog front end (AFE) while the DTSA is acting as a load for the continuous time linear equalizer (CTLE).

Once we used noise simulations to get the input referred voltage noise of the comparator, we can then aggregate the comparator's noise from previous simulations with the AFE's noise to compute the total rms noise comparator's input. We also use the transient response of the circuit to assure high fidelity for the eye diagram at the input of comparator. The input/output curves, and formulas used to measure eye's fidelity are shown in figure 3.7a and 3.7b.

We estimate eye height and eye thickness ratio, and specify a constraint on them to describe the quality of eye diagram for a given input sensitivity. Using BER of 10^{-12} we can compute the required eye height at the input of the comparator using equation 3.1 and compare it against the actual eye height. For the optimization objective we can put a constraint on the relative difference of the actual eye height and the required eye height (i.e. actual eye height should be 10% larger than the minimum required eye height. We call this percentage eye margin).

$$eye_h_{min} = 9\sigma_{noise} + \text{Residual Offset} + \text{DTSA sensitivity} \quad (3.1)$$

For sensitivity we use the V_{in} from the overdrive test recovery in previous tests. There will also be a component mismatch offset which can significantly reduced with a systematic offset

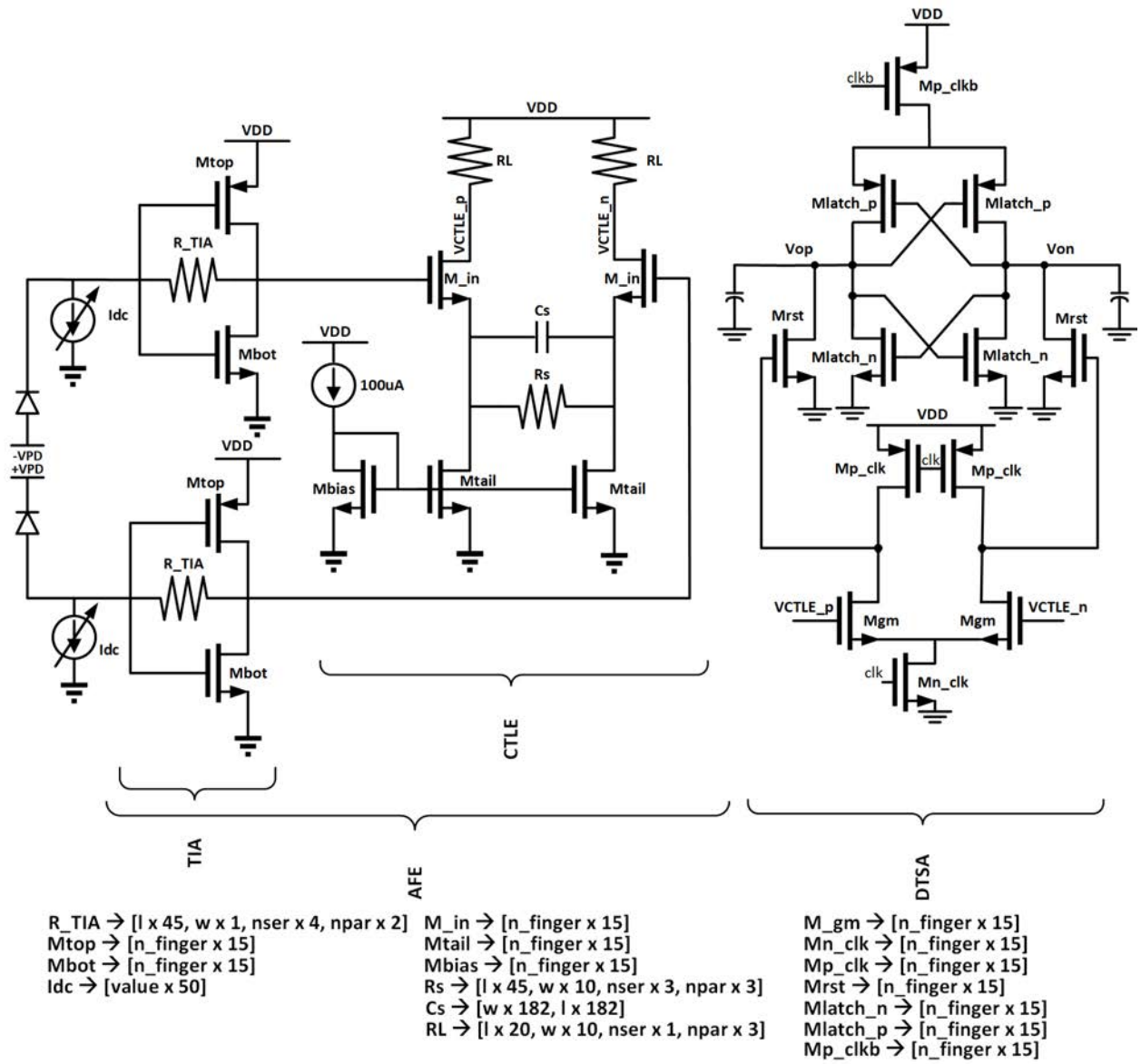
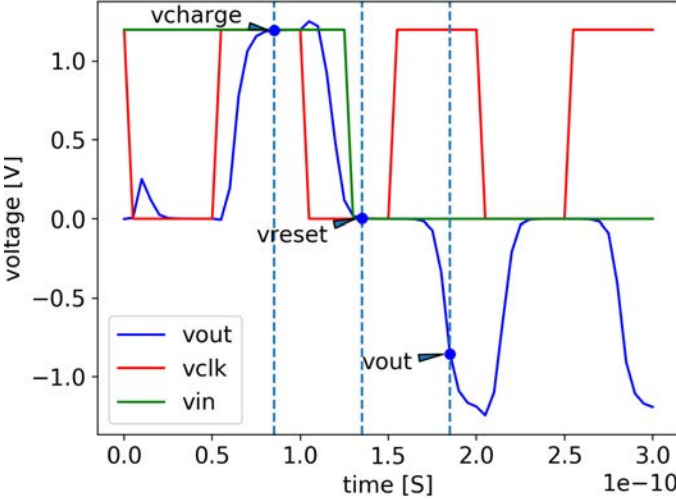
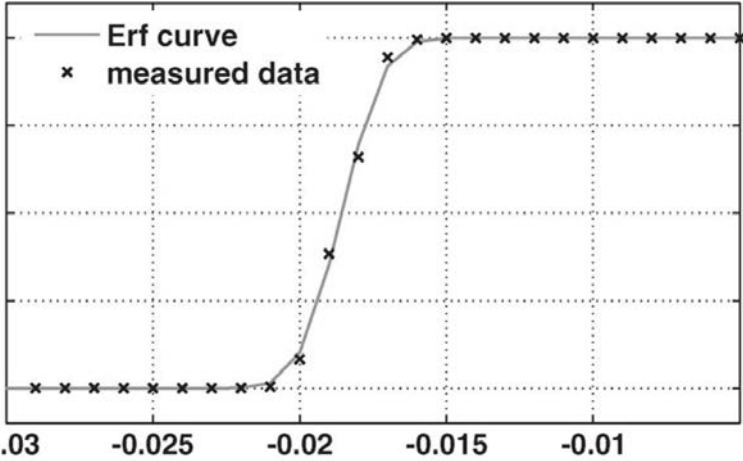


Figure 3.5: Optical receiver schematic

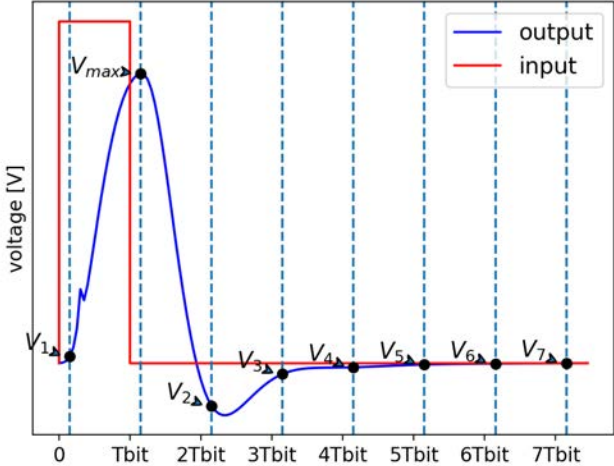


(a)

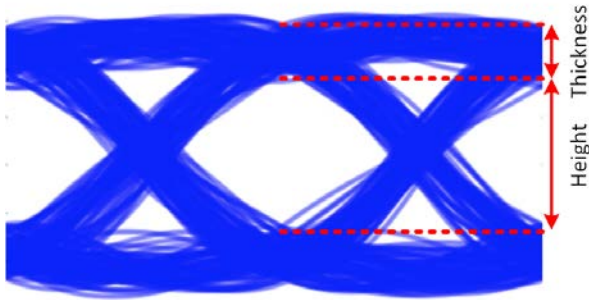


(b)

Figure 3.6: a) Overdrive test recovery simulation curves for DTSA b) Probability of outputting a one vs. V_{in} . We can use the cumulative density function of a Gaussian to estimate the standard deviation of the noise



(a)



$$eye_height = V_{max} - \sum_{i=1}^{\infty} |V_i|$$
$$thickness = \sum_{i=1}^{\infty} |V_i|$$

(b)

Figure 3.7: a) Input and output signals for measuring the eye height and thickness, the input is a small signal pulse with an amplitude of target input sensitivity, and with a width of T_{bit} for the target data rate. The output curve is sampled at time instances shown relative to the main cursor (maximum of output) b) Equations used for estimating eye height and thickness to express the fidelity of eye diagram

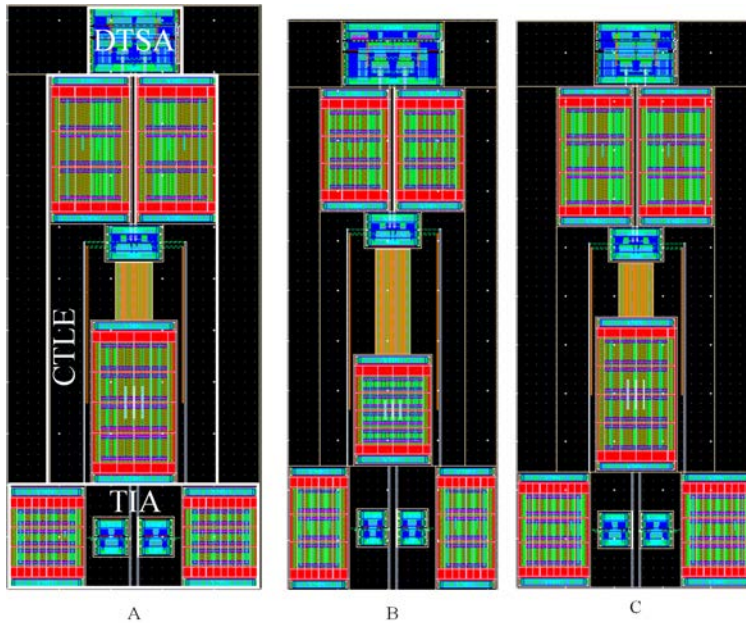


Figure 3.8: Sample layouts of the optical link receiver circuit with different cost values of 0, 0.1, and 0.2 for A, B, and C, respectively.

cancellation scheme. The offset cannot, however, be fully eliminated, so the residual offset will also be considered (i.e. 1mV). We also run a common mode ac simulation to ensure that the tail transistors providing bias currents are operating in saturation by specifying a reasonable minimum common mode rejection ratio requirement. For one instance, this whole process takes about 200 seconds on our compute servers. We can then compute the cost of each design as specified by equation 2.1. Therefore, being sample efficient in terms of learning and optimizing is vital, as discussed in section 3.1.

In terms of layout generator search space, each resistance drawn in Figure 3.5 has unit length, unit width, number of series units, and number of parallel units. The CTLE’s capacitor has width and length, and each transistor has number of fins and number of fingers that need to be determined. We fixed number of fins to simplify the search space. The cardinality of each design parameter is written in Figure 3.5. In total, the design example has a 26 dimensional exploration space with size of 2.8×10^{30} .

Figure 3.8 and Table 3.4 show the layout and performance of the solution for designs found with cost of 0 (satisfying all specs), 0.1, and 0.2, respectively. The first design solution was found using 435 simulations equivalent to 27 hours of run time. This number includes generating the initial population which consisted of 150 designs with the best cost function of 2.5. During the process the DNN was queried 77487 times in total and only 285 of those were simulated, representing around 300x sample compression efficiency. From the total run time 1.6 hours were spent on training and almost 2.1 hours were spent on querying the DNN.

Table 3.4: Design Performance for optical receiver design for $C_{PD} = 20fF$, $I_{min} = 3\mu A$, Data Rate = $10Gbit/s$

	Requirement	A (cost = 0)	B (cost = 0.1)	C (cost = 0.2)
Thickness ratio	<10%	6%	3.3%	8.5%
Eye margin @ I_{min}	>10%	10.2%	15.1%	10.4%
CMRR	>3	4.77	5.51	4.5
vcharge	>0.95VDD	VDD	VDD	VDD
vreset	<1 mV	20 μV	52 μV	21 μV
vout	<-0.9VDD	-0.91VDD	-0.88VDD	-0.87VDD
Total Noise	<5 mV	2.9 mV	3.2 mV	3.05 mV
Total Ibias	<10 mA	6.2 mA	4.04 mA	6.3 mA

Chapter 4

Conclusion and Future Work

In this report we have proposed a new sample-efficient evolutionary-based optimization algorithm for designing analog circuits using layout generators. We showed that the algorithm can be used in designing a variety of real, practical analog/mixed signal circuits with different applications regardless of size and complexity, as long as the verification procedure is properly defined. In author's mind the following items are the natural directions for this project's future:

1. We have to setup a quantitative benchmark for measuring performance of our agent against human experts and the Oracle. It is not clear how well the training process is taking place and we would like to see an unbiased performance metric for comparing different DNN models deployed in this BagNet.
2. With BAG's evaluation engine, debugging based on post-extracted simulation is very time consuming, and cumbersome. For instance we have to make sure that the verification methodology does not have any bugs. In this work, we check them using random generation and evaluation of a large number of designs and then, go through them manually to see if every thing makes sense. By making sense, we mean that, sorting designs by cost produces a consistent profile of good to bad designs from designer's stand point. This is really inconvenient, since if there is any issues somewhere in the code we have to run evaluation for a large number of designs again. For simulation BAG's API is limited. We need to speed up the simulation process in BAG, so that it gets more convenient to debug.
3. Continuing on previous item's argument, it would be interesting to see if we train something based on schematic can we transfer this knowledge to layout with the same parameters and speed up convergence of the algorithm? This way we can start by optimizing a schematic and then fine-tuning the layout.

The author believes that recent advancement in artificial intelligence and computational hardware has opened many new doors in automation tasks. There are many creative ideas that could be explored that could disrupt the CAD industry for analog as well as digital.

Appendix A

Framework Implementation ¹

In this chapter we will discuss some of the details of the framework we developed.

A.1 Overview of Framework Structure

Figure A.1 illustrates how different modules interact with each other in this framework. There is a central agent module which handles the flow of algorithm ². It constructs other modules and calls their appropriate methods as necessary. The following sections will discuss the details of each module.

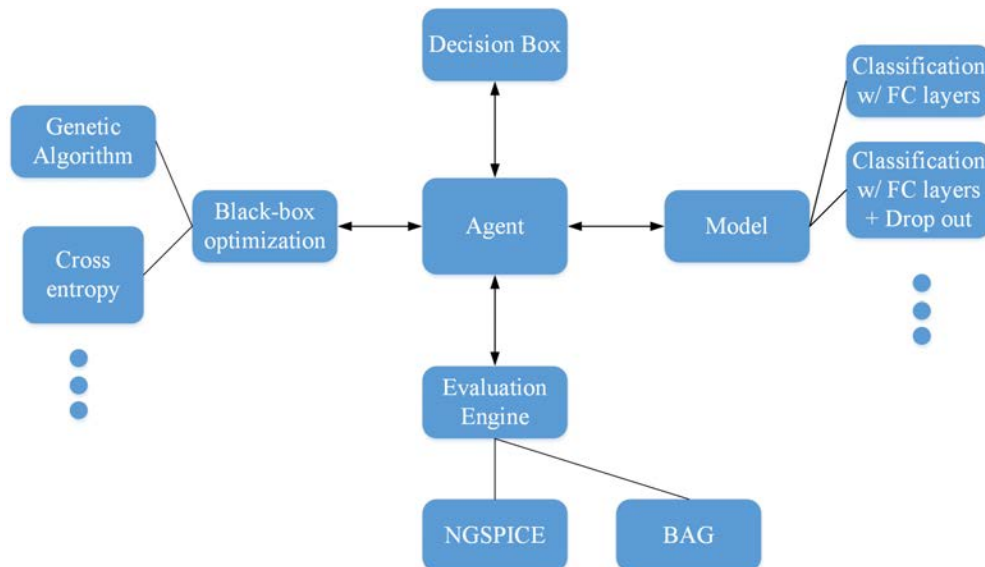


Figure A.1: The overview of framework modules

¹The code is available on [github](#)

A.2 Evaluation Engine

This module is responsible for running and querying the evaluation core. It also handles parallel evaluation. The module itself is initialized by a specification file that has all the information about the parameters to optimize, their range, the circuit specification, and how to evaluate the cost function. The interface functions are the following:

```
def generate_data_set(n, evaluate)
    """
    Generates a data set of size n and if evaluate is True runs the
    ↪ evaluation for each of them. Returns a list of Design objects with
    ↪ specification populated for each design.
    """
def evaluate(design_list)
    """
    Evaluates a list of designs. Returns a list of specification
    ↪ dictionaries according to the order of the designs that were passed
    ↪ in.
    """
def compute_penalty(spec_num, spec_kwrtd)
    """
    Computes the penalty of specification 'spec_kwrtd' that has a value of
    ↪ 'spec_num' according to threshold which it should be above or below.
    """
```

There are two different evaluation cores implemented in this framework: First one is an NGSPICE evaluation engine. NGSPICE is an open source SPICE-like simulator that can be used on any machine without license requirements. It is fast since it does only schematic simulation and reading the data is easy. Second one is BAG integrated evaluation engine which can generate layout, schematic, run LVS and RC extraction, and finally run simulations with post extracted results. This evaluation engine is expensive in run-time.

A.3 Model

This module represents the neural network model that has been implemented in the back end. It needs three interface functions implemented:

```
def get_train_valid_ds(*args, **kwargs)
    """
    Returns training and validation datasets with their proper output
    ↪ labels for the model in numpy arrays formats
```



```

    """
def train(*args, **kwargs)
    """
    Trains the model the way it should be trained, this should be
    ↪ implemented compatible with get_train_valid_ds and model parameters
    """
def query(**kwargs)
    """
    Inference part of the model. Input should be input parameters of the
    ↪ DNN and output is the predicted labels
    """

```

We have tried two flavors of models. One is simple classification model with fully connected layers only (three layers of size 20 for feature extraction layer and three layers of size 20 for each specification classifier). Another one is the same model with drop out layers in between FC layers to decrease over-fitting and to get better estimates in low data regime. Drop out layers were tried with dropping probability of 20 percent (keep_prob=0.8). The drop out model tends to work better qualitatively, but as was discussed in chapter [4](#), in future venues more work needs to be done for quantifying the performance.

A.4 Black-box Optimization

This module handles all the evolutionary operations needed for the optimization framework. It has the following interface:

```

def get_next_generation_candidates(parents1, parents2)
    """
    Generates candidates by doing some evolutionary operations on one
    ↪ individual drawn from each of 'parents1' and 'parents2'.
    """
def prepare_for_generation(db, n)
    """
    Prepares the evolutionary algorithm for generating samples. An
    ↪ example is fitting distribution parameters on the data in cross
    ↪ entropy optimization.
    """

```

We implemented two flavors of evolutionary operations. One is a custom evolutionary strategy that is outlined below:

```

def get_next_generation_candidates(parents1, parents2):
    if len(parents1) == 0:
        parents1 = parents2
    assert (self.cxpb + self.mutpb) <= 1.0, (
        "The sum of the crossover and mutation probabilities must be
        ↪ smaller "
        "or equal to 1.0.")
    op_choice = random.random()
    offsprings = []
    if op_choice <= self.cxpb:                # Apply crossover
        # randomly select one individual from parents1 and another from
        ↪ parents2
        parent1, parent2 = self._select_parents_from_two_pops(parents1,
            ↪ parents2)
        ind1 = parent1.copy()
        ind2 = parent2.copy()
        # mates the individuals to create new offspring, the mating is a
        ↪ mixture of blend and 2 point crossover}
        ind1, ind2 = self._mate(ind1, ind2, low=self.lows, up=self.ups)
        offsprings += [ind1, ind2]
    elif op_choice < self.cxpb + self.mutpb: # Apply mutation
        # randomly select one individual from parents1
        parent1 = self._select_for_mut_based_on_order(parents1)
        # randomly perturb it an clip it to lower and upper bounds
        new_ind, = self._mutate(new_ind, low=self.lows, up=self.ups)
        offsprings.append(new_ind)

    return offsprings

```

The other one is the cross entropy method (CEM) [4]. In this stochastic optimization, at each `get_next_generation_candidates` call, we produce each candidate by sampling the parameters from a normal distribution: $\theta_i \sim N(\mu_k, \Sigma_k)$. For `prepare_for_generation`, we re-fit a new mean and variance to the top designs in the population.

A.5 Decision Box

This module handles all the heuristic-based decisions in the algorithm, such as determining which designs to accept based on DNN's output, whether the convergence condition is met, and getting parents and the reference design from population.

Bibliography

- [1] Vincentelli et al. “Support vector machines for analog circuit performance representation”. In: (2003), pp. 964–969.
- [2] G. Alpaydin, S. Balkir, and G. Dunder. “An evolutionary approach to automatic synthesis of high-performance analog integrated circuits”. In: *IEEE Transactions on Evolutionary Computation* 7.3 (June 2003), pp. 240–252.
- [3] Manuel F. M. Barros, Jorge M. C. Guilherme, and Nuno C. G. Horta. *Analog Circuits and Systems Optimization based on Evolutionary Computation Techniques*. Vol. 294. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2010.
- [4] Pieter-Tjerk de Boer et al. “A Tutorial on the Cross-Entropy Method”. In: *Annals of Operations Research* 134.1 (Feb. 2005), pp. 19–67.
- [5] Eric Chang et al. “BAG2: A process-portable framework for generator-based AMS circuit design”. In: *2018 IEEE Custom Integrated Circuits Conference (CICC)*. San Diego, CA: IEEE, Apr. 2018, pp. 1–8.
- [6] Marc G R Degrauwe et al. “IDAC: An Interactive Design Tool for Analog CMOS Circuits”. In: ().
- [7] Yarin Gal and Zoubin Ghahramani. “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”. In: (June 2015). URL: <http://arxiv.org/abs/1506.02142>.
- [8] G. Gielen, P. Wambacq, and W.M. Sansen. “Symbolic analysis methods and applications for analog circuits: a tutorial overview”. In: *Proceedings of the IEEE* 82.2 (Feb. 1994), pp. 287–304.
- [9] G. Gielen et al. “An analogue module generator for mixed analogue/digital asic design”. In: *International Journal of Circuit Theory and Applications* 23.4 (July 1995), pp. 269–283.
- [10] R. Harjani, R.A. Rutenbar, and L.R. Carley. “OASYS: a framework for analog circuit synthesis”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 8.12 (Dec. 1989), pp. 1247–1266.
- [11] J.P. Harvey, M.I. Elmasry, and B. Leung. “STAIC: an interactive framework for synthesizing CMOS and BiCMOS analog circuits”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 11.11 (Nov. 1992), pp. 1402–1417.

- [12] M.delM. Hershenson, S.P. Boyd, and T.H. Lee. “Optimal design of a CMOS op-amp via geometric programming”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 20 (Jan. 2001).
- [13] H.Y. Koh, C.H. Sequin, and P.R. Gray. “OPASYN: a compiler for CMOS operational amplifiers”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 9.2 (Feb. 1990), pp. 113–125.
- [14] Michael Krasnicki et al. “MAELSTROM: Efficient Simulation-Based Synthesis for Custom Analog Cells”. In: ().
- [15] Wim Kruiskamp and Domine Leenaerts. “DARWIN: CMOS opamp Synthesis by means of a Genetic Algorithm”. In: ().
- [16] Radford M. Neal. *Bayesian Learning for Neural Networks*. Ed. by P. Bickel et al. Vol. 118. Lecture Notes in Statistics. New York, NY: Springer New York.
- [17] W. Nye et al. “DELIGHT.SPICE: an optimization-based system for the design of integrated circuits”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 7 (Apr. 1988), pp. 501–519.
- [18] E.S. Ochotta, R.A. Rutenbar, and L.R. Carley. “Synthesis of high-performance analog circuits in ASTRX/OBLX”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 15 (Mar. 1996), pp. 273–294.
- [19] Rodney Phelps and James R Hellums. “Anaconda: Simulation-Based Synthesis of Analog Circuits Via Stochastic Pattern Search”. In: 19 ().
- [20] Pankaj P. Prajapati and Mihir V. Shah. “Two stage CMOS operational amplifier design using particle swarm optimization algorithm”. In: IEEE, Dec. 2015.
- [21] Stephane Ross, Geoffrey J Gordon, and J Andrew Bagnell. “A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning”. In: ().
- [22] Rob A. Rutenbar, Georges G. E. Gielen, and Brian A. Antao. *Computer-Aided Design of Analog Integrated Circuits and Systems*. IEEE, 2002.
- [23] A. Torralba, J. Chavez, and L.G. Franquelo. “FASY: a fuzzy-logic based tool for analog synthesis”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 15 (July 1996), pp. 705–715.
- [24] P. Wambacq et al. “Efficient symbolic computation of approximated small-signal characteristics of analog integrated circuits”. In: *IEEE Journal of Solid-State Circuits* 30.3 (Mar. 1995), pp. 327–330.
- [25] G.A Wolfe. “Performance macro-modeling techniques for fast analog circuit synthesis. Ph.D. dissertation, Dept. of Electrical and Computer Engineering and Computer Science, College of Engineering, University of Cincinnati, USA”. PhD thesis. 1999.