UNIVERSITY OF CALIFORNIA,
IRVINE

Design and Analysis of Low Noise Amplifiers in Ngspice with a Perspective on Machine Learning Approaches

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Electrical and Computer Engineering

by

Pooya Khajeh

Thesis Committee:
Assistant Professor Hamidreza Aghasi, Chair
Professor Michael M. Green
Assistant Professor Roy Fox
Assistant Professor Mohsen Imani

2023

DEDICATION

To my parents, my siblings, and my friends, for their unfaltering love and support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I wish to express my gratitude to my adviser, Assistant Professor Hamidreza Aghasi for his continuous guidance and motivation. I would also like to extend my thanks to Assistant Professor Roy Fox whose expertise and insights greatly enriched my work on "Learning to Design Analog Circuits to Meet Threshold Specifications".

Special thanks to Dmitrii Krylov, Tongkai Liu, Junhan Ouyang, Thomas Reeves, and Hiba Ajmal for their contribution to this work.

Also, I would like to thank Mohammad Oveisi, who has been a constant source of encouragement and companionship throughout my time in graduate school. Having him by my side made my journey all the more memorable.

The text of this thesis is an adaptation of the material as it appears in "Learning to Design Analog Circuits to Meet Threshold Specifications", the Proceedings of the 40th International Conference on Machine Learning, used with permission from ML Research Press. The co-authors listed in this publication are Dmitrii Krylov, Junhan Ouyang, Thomas Reeves, Tongkai Liu, Hiba Ajaml, Hamidreza Aghasi, and Roy Fox. Hamidreza Aghasi directed and supervised research which forms the basis for the thesis.

ABSTRACT OF THE THESIS

Design and Analysis of Low Noise Amplifiers in Ngspice with a Perspective on Machine Learning Approaches

By

Pooya Khajeh

Master of Science in Electrical and Computer Engineering

University of California, Irvine, 2023

Assistant Professor Hamidreza Aghasi, Chair

The primary objective of this thesis is to explore the design and simulation of Low Noise Amplifiers using Ngspice. Additionally, it delves into a machine learning approach aimed at learning design based on threshold specifications, with Ngspice serving as the simulation engine.

Chapter 1 introduces the background and underscores the significance of the research. Chapter 2 delves into Ngspice, highlighting its applications in analog and RF design. Chapter 3 is dedicated to the design and simulation of a Low-Noise Amplifier (LNA) utilizing Ngspice. Chapter 4 revisits our prior work titled "Learning to Design Analog Circuits to Meet Threshold Specifications", placing special emphasis on the LNA circuit incorporated within.

Chapter 1

Introduction

RF (radio frequency) and analog circuit design are of great importance in modern electronics. Analog circuits are essential for many electronic systems, enabling functions such as amplification, filtering, and signal processing. Meanwhile, RF circuits are critical for high-performance electronic systems such as wireless communication, high-speed data converters, and precision measurement systems. These systems require circuits with high linearity, low noise, and high dynamic range, which can only be achieved through careful RF and analog circuit design. Moreover, power-efficient analog and RF circuits are essential for modern portable electronic devices, such as smartphones and wearable, to conserve battery life and enable wireless communication. Furthermore, continuous innovation in RF and analog circuit design is necessary to drive new technologies and applications, such as 5G wireless communication, autonomous vehicles, and internet of things (IoT) devices. Overall, RF and analog circuit design plays a critical role in high-performance, power-efficient, and innovative electronic products and technologies[1] [21].

It is important to point out that Analog and RF design is a topic of great interest, as is the case with any important topic, but it can also be difficult and Challenging. It is important

for the engineer to possess a thorough understanding of Electromagnetic Theory, Signal Processing, and Circuit Design principles to be able to design these systems effectively. In addition, designing can be burdensome for designers because of a variety of factors, including:

The designs of RF and analog ICs are often quite complex due to the large number of components involved and interactions between them. As a result of this complexity, it can be difficult for designers to optimize the performance of their circuits while simultaneously meeting power consumption, noise, and other specifications. RF and analog circuits are usually sensitive to small variations in component values, temperature, and other environmental factors. It may be challenging for circuit designers to ensure that their circuits will operate reliably in a wide range of operating conditions as a result of this sensitivity. The design of RF and analog integrated circuits is time-consuming, as designers are often required to perform numerous simulations and tests in order to ensure that their circuits are functioning properly. The cost of errors in RF and analog IC designs can be very high, as even small errors may have a significant effect on performance or even result in circuit failure.

Significance of a precise and reliable simulation engine , is paramount in ensuring accuracy and efficiency in electrical engineering and circuit design endeavors. In the next chapter one of these functional and pivotal simulators (Ngspice) is studied.

Beyond the precision of a simulation engine, tools that streamline and expedite the simulation process are invaluable. Machine learning serves as one such tool, offering potential advancements in efficiency and effectiveness[2]. Machine learning as an assisting tool can minimize the time spend on a design while making the design more effective and also prevent human errors.Utilizing machine learning as an auxiliary tool can significantly reduce design time, enhance design effectiveness, and mitigate human errors. chapter 4 discusses a novel method for designing analog circuits by deploying threshold specification technique[12].

Chapter 2

Ngspice

Ngspice is an open-source mixed-level/mixed-signal electronic circuit simulator. It is a successor of the famous SPICE simulator (Simulation Program with Integrated Circuit Emphasis), originally developed at the University of California, Berkeley, in the late 1970s.

NGSpice has gained popularity for several reasons. It is open-source and freely available for anyone to use, modify, and develop. For students and researchers who lack access to commercial simulation software, NGSpice is an attractive option. This community also provides comprehensive documentation, and tutorials making it easier to get the most out of the software. Being open-source its development is drived by large community of users and developers. Thus, continuously updates, bug fixes and enhanced features are being added. Also, given its open-source nature and versatility, it can be widely used in academia for teaching and research purposes.

Aside from that, Ngspice has attracted the attention of circuit designers due to its accuracy, versatility, compatibility with SPICE and other tools. Simulations are provided using the SPICE engine, which is a reliable source of information. It supports AC, DC, transient, noise, and more simulations. Ngspice is a mixed-level/mixed-signal simulator capable of

simulating analog, mixed-signal, and digital circuits. This makes it suitable for a wide range of applications from simple analog circuits to complex integrated circuits. It is compatible with SPICE syntax, so a SPICE user can simply and quickly exploit Ngspice features. Additionally, it can be integrated with other commercial and open-source tools such as KiCad for Printed Circuit Board design, GNU Octave for numerical computations, or different graphical interfaces for post-processing simulation data.

In subsequent chapters, we delve into the design of a Low Noise Amplifier (LNA) as one of the most vital and popular components in Recievers. Additionally, we explore design methodologies employing machine learning approaches, all implemented and analyzed using Ngspice as the simulation engine.

Chapter 3

Low Noise Amplifier

3.1 Background

In many receiver structures, low noise amplifiers are the first active block used, and hence are of significant importance and . Based on Friis equation shown [8] in Equation 1, Gain and noise of the Whole receiver chain can be impacted by first stages characteristics.

$$F_{total} = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} + ... + \frac{F_N - 1}{G_1 G_2 ... G_{N-1}} \tag{3.1}$$

Hence LNA plays an important role in general performance of a receiver, and as a result is of great importance. Consequently, ample research has been undertaken on LNAs, resulting in the proposal of numerous circuitries and topologies. In this work a single ended cascode inductive degeneration LNA is designed and studied. The common structure of this LNA is shown in fig.3.2

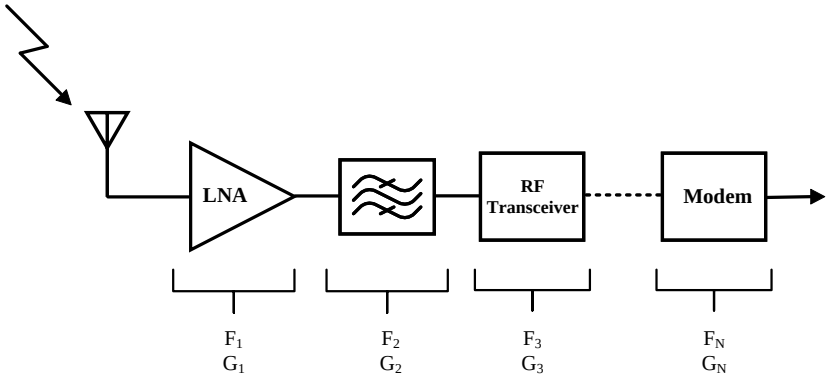

Figure 3.1: General block diagram of receiver

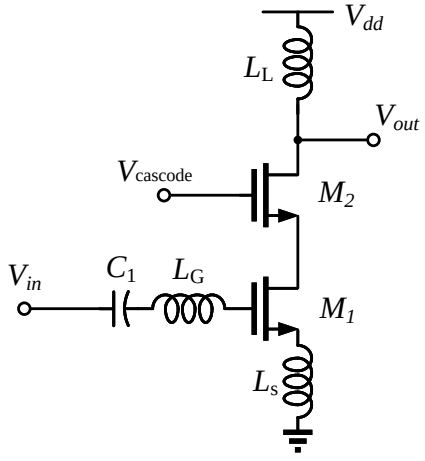

Figure 3.2: General block diagram of a Cascode LNA

3.2 Design

Using an impedance match method for the input of the LNA, and optimizing the width of the cascade transistors, the input network is designed as shown in fig.3.3. According to the figure, the source resistance is considered 50 ohms. The value of C_1 as the DC blocking cap is chosen to prevent deviations in gate to source bias of transistor.M_1. Transistor M_3 Serves as the current mirror for the transistor M_1 and has a small fraction of M_1's width to limit the power overhead. [14]. Resistors R_1 is serving as the identifier of the current value passing through M_3. R_2 is chosen large enough to isolate the biasing circuit impedance, and have negligible noise current[24]

3.2.1 Matching and S-parameters

To draw the impedance smith chart of the input resistance first S-parameters obtained using the commands in Ngspice shown in fig.3.4. In Figure you can see that the S-parameters are measured and extracted for frequencies between 1 MHz and 10 GHz, with arbitrary linear stepping of 100 points being used. Subsequently, the s-parameters are extracted in the format of `S_n_n`, which `n` indicated the number of RF ports used. To monitor the input matching S_{11} is calculated in dB and it is plotted as shown in fig.3.5. As shown for frequencies with $S_{11} < -10dB$ a bandwidth of almost equal to 850 MHz for S_{11} is achieved.

To obtain the input impedance, S-parameters are converted to Z_{11} by using [20]:

$$Z_{11} = Z_0 \frac{(1 + S_{11})(1 - S_{22}) + S_{12}S_{21}}{(1 - S_{11})(1 - S_{22}) - S_{12}S_{21}} \tag{3.2}$$

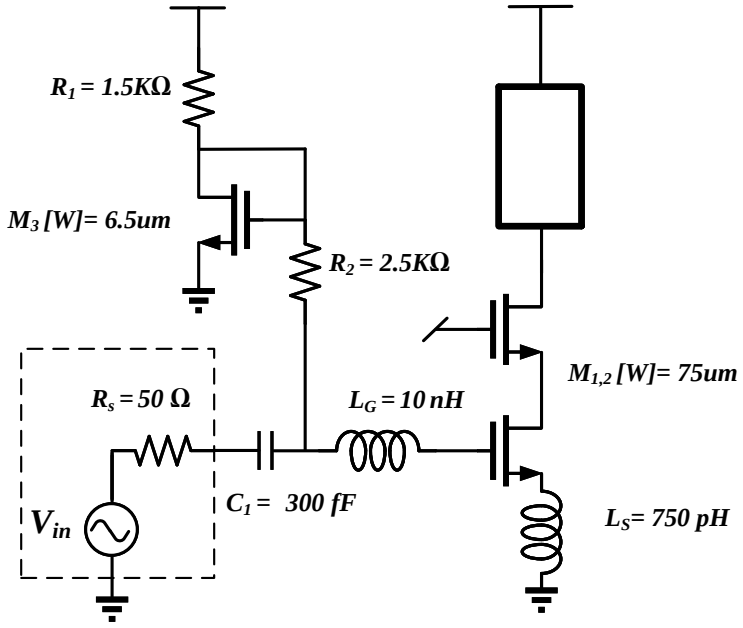

Figure 3.3: Input marching network and biasing of the LNA

```
Sp lin 100 1M 8G
let S11=S_1_1
let S12=S_1_2
let S21=S_2_1
let S22=S_2_2
let s11db=20*log(S11)
plot S11db
```

Figure 3.4: Ngspice code for S-parameters analysis

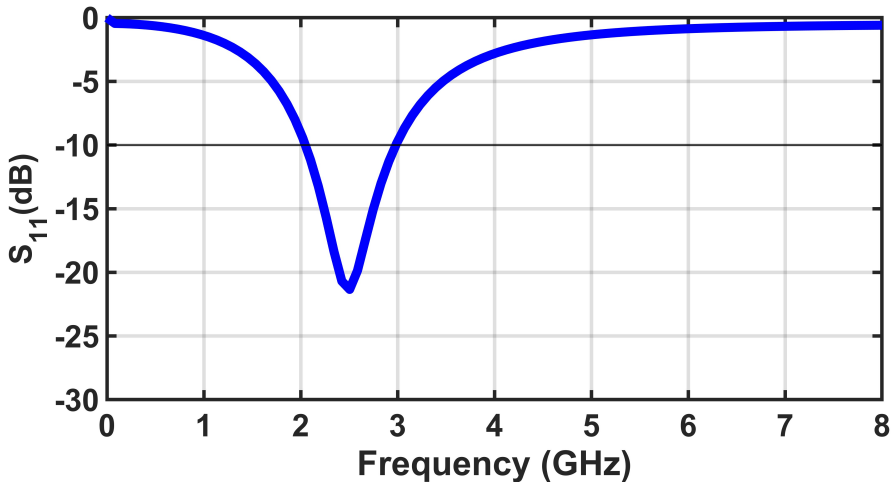

Figure 3.5: $S_{11}(dB)$

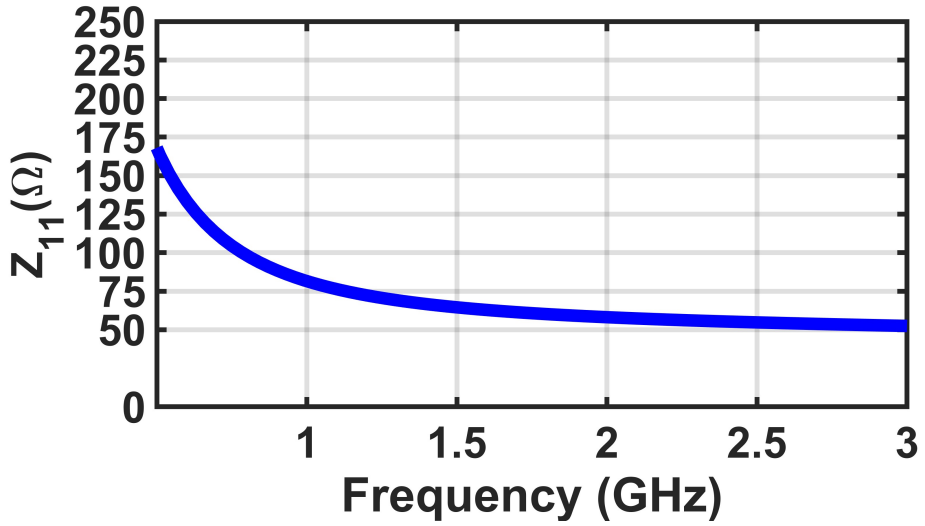

Figure 3.6: Z_{11} Magnitude

```
let Ls_start=100p
let Ls_step=50p
let Ls_stop=2n
let Ls_test=ls_start
while Ls_test le ls-start
alter Ls = Ls_test
sp lin 100 1M 3G
let Z11(magnitude)= mag(50*((1+S11)*(1-S22)+S12*S21)/((1-S11)*(1-S22)-S12*S21))
let Z11=z11(magnitude)
plot Z11
```

Figure 3.7: Ngspice code for sweeping variable L_s

$Z_{11}$is achieved.As can be seen in Fig. 3.8, the calculated Z11 for the proposed input matching network has been shown. The best match to 50ohm-impedance occurs around f=2.5GHz, as can be seen from the image.

As it stands, the value of the degeneration inductor, L_s, which is responsible for the real part of the input resistance, has been swept over a range of possible values, and at the end, the value of 750 pH was determined to be the most appropriate value. The commands for finding the optimum value of L_s is shown in fig.3.7.

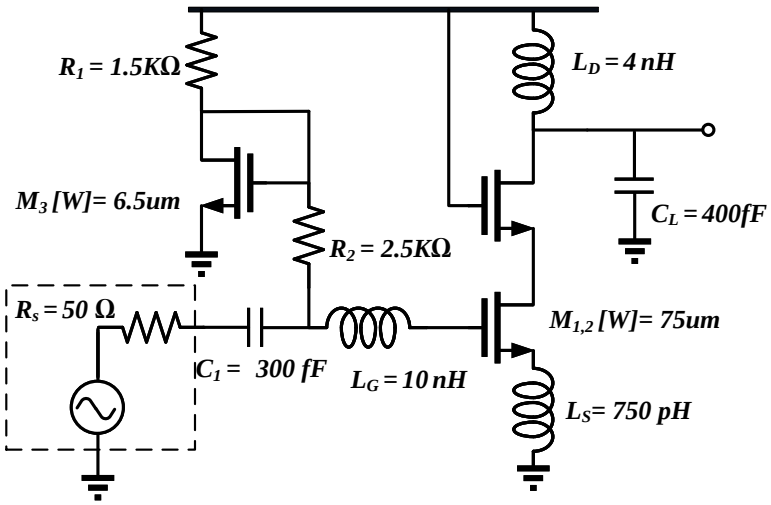

Figure 3.8: Final LNA Schematic

```
V1 net7 0 dc 0 ac 1 portnum 1 z0 50
AC dec 100 10k 10G
plot vdb(out) xlog
```

Figure 3.9: AC analysis commands

3.2.2 AC Analysis and Voltage Gain

To have a higher gain at the desirable frequency, and providing a better band pass filtering, L_D is implemented to resonate with total capacitance at the drain of M_2. The resonate frequency as the output is arbitrarily chosen the same or close enough to the input resonance frequency to have a narrow response[14]. A load capacitor of C_L is also used at the output for better bandwidth control. The final Schematic for the proposed LNA is shown in fig.

The Magnitude Bode plot for this LNA is achieved by running AC analysis as shown in fig.3.9. Defining an input voltage source with AC value (in this example, 1) is necessary. An AC analysis was conducted in decade increments, beginning at 100 kHz and concluding at 10 GHz. Bode plot for the magnitude of the gain is shown in fig.3.10.

As illustrated in Fig. 3.10, the value of L_D is chosen to provide the maximum gain magnitude

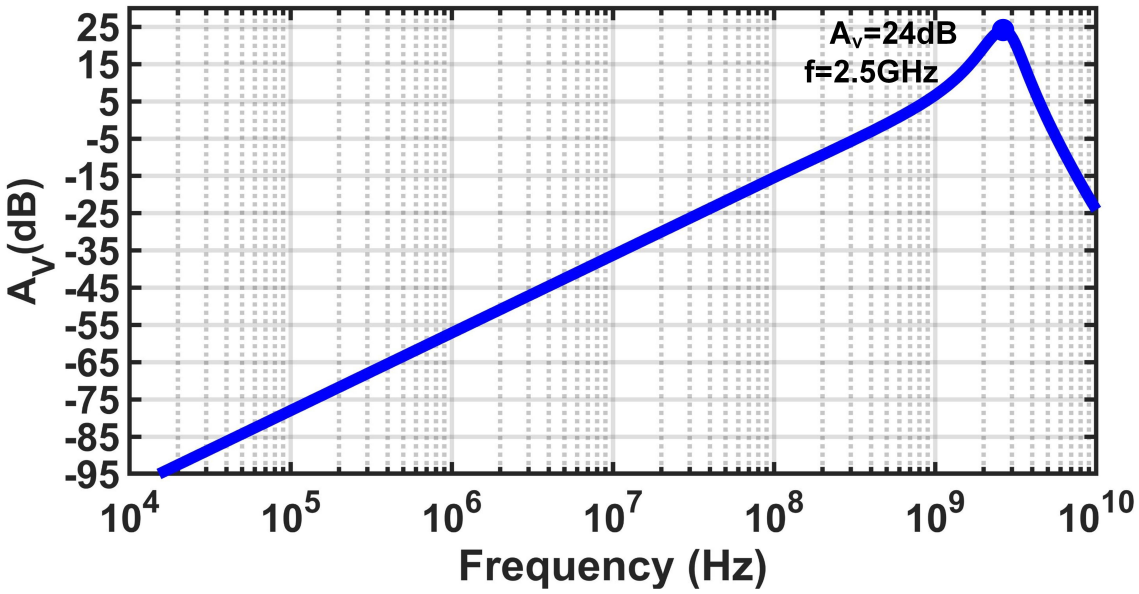

Figure 3.10: Gain Magnitude Bode Plot

at the input resonant frequency of 2.5GHz.

3.2.3 Stability

A crucial aspect of LNA design is ensuring stability. Various methods exist to assess stability, with the K and Delta method being one such approach[6], Which states For an amplifier to remain stable, it requires that:

$$K > 1 \tag{3.3}$$

and

$$|\Delta| < 1 \tag{3.4}$$

Which K and Δ are[9]:

$$K = \frac{1 - |S_{11}|^2 - |S22|^2 + |\Delta|^2}{2|S_{12}S_{21}|} \tag{3.5}$$

and

$$\Delta = S_{11}S_{22} - S_{12}S_{21} \tag{3.6}$$

To ensure the stability, K and $|\Delta|$ are simulated and verified as shown respectively in fig.3.11 and fig.3.12. As it is shown in the plots, K is bigger than one and $|\Delta|$ is smaller than one over the whole frequency spectrum. Hence, the amplifier is stable.

3.2.4 Noise Figure

As the name suggests, the primary function of an LNA is to amplify weak signals without adding significant noise. The noise figure quantifies how much noise the LNA adds to the signal it's amplifying. LNA NF is a vital performance metric since , it is directly added to that of the reciever[1]. In Ngspice an s-parameter analysis, with adding a "Do Noise Option" with capture Noise Figure in dB. The NF(dB) for the proposed LNA is shown in fig.3.13. As shown the minimum NF is available around the favorable frequency of 2.5GHz.

3.2.5 Power Gain

In RF amplifier design and analysis, a pivotal performance metric is the power gain. Sophisticated simulation tools such as Cadence offer direct capabilities to compute power gains.

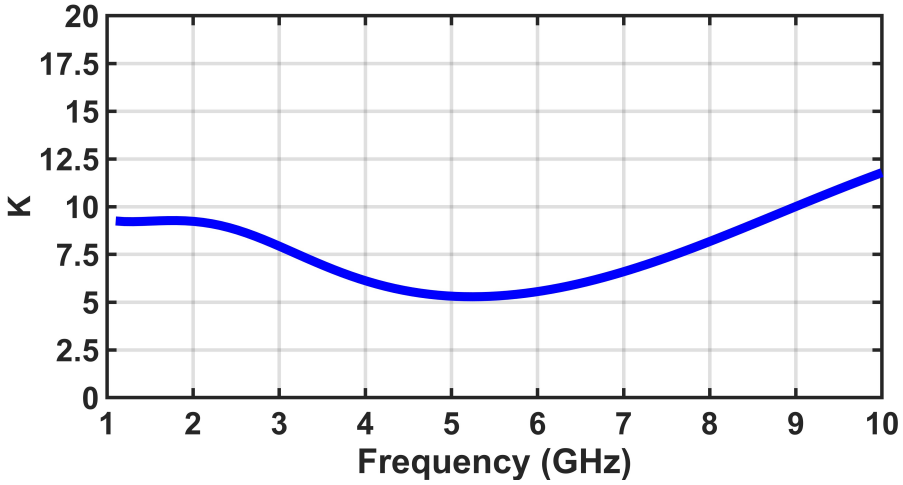

Figure 3.11: K

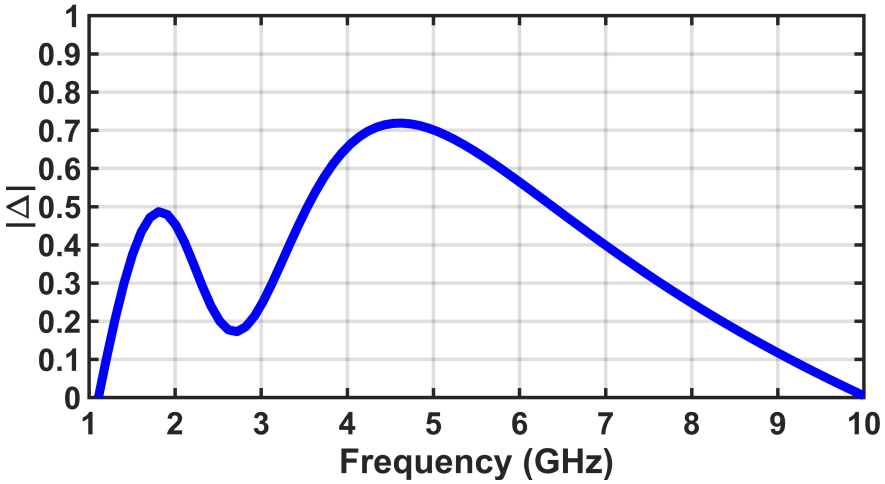

Figure 3.12: $|\Delta|$

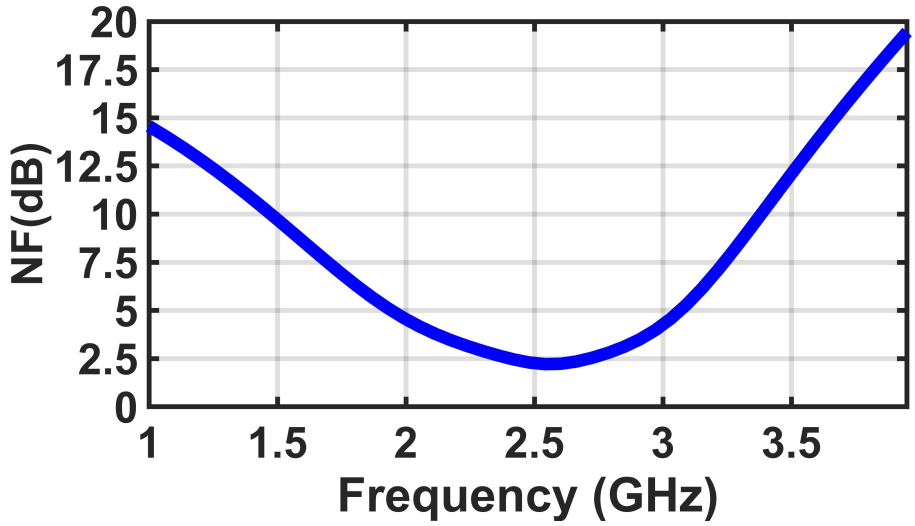

Figure 3.13: Noise Figure(dB)

However, open-source platforms like Ngspice lack this direct feature. Nevertheless, under certain specific conditions, Ngspice users can still navigate this limitation.

A crucial premise for achieving this is to ensure the amplifier's source and load reflection coefficients are matched to the conjugate values of S_{11} and S_{22}, respectively. Meaning, for a negligble S_{12} for the case of unilateral transducer gain we can say:[27]:

$$G_{TU} = |S_{21}|^2 \frac{1 - |\Gamma_S|^2}{|1 - S_{11}\Gamma_S|^2} \frac{1 - |\Gamma_L|^2}{|1 - S_{22}\Gamma_L|^2} \tag{3.7}$$

Hence if :

$$\Gamma_S = S_{11}^* \tag{3.8}$$

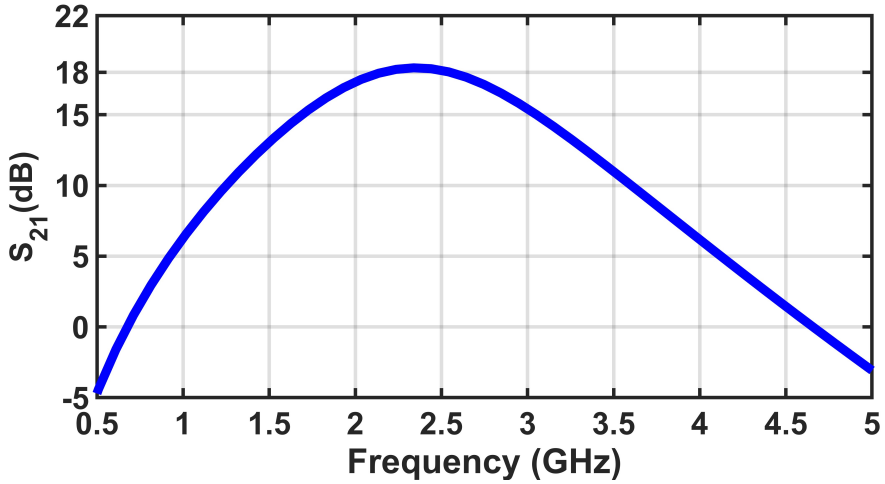

Figure 3.14: Proposed LNA S_{21}

```
let Transducer_Power_Gain= (vecmax (S21))^2
meas sp  Resonant_Frequency WHEN S21 = vecmax (S21)
```

Figure 3.15: Commands to extract the Unilateral Transducer Power Gain and the resonant frequency

$$\Gamma_L = S_{22}^* \tag{3.9}$$

The we have:

$$G_{TU} = G_O = |S_{21}|^2 \tag{3.10}$$

or

$$G_{TU}(db) = 20log(|S_{21}|) \tag{3.11}$$

```
op
let iDC= @M1[id]
let DC_Power_Consumption= Vdd*iDc
Print DC_Power_Consumption
```

Figure 3.16: DC Analysis command

3.2.6 DC Power Consumption

Ngspice offers both DC and OP (Operating Point) Analysis, which can be employed to compute the DC power consumption by multiplying the current with the headroom voltage. By assessing the current flowing through the transistor M_1, we can achieve an approximation that closely represents the actual DC current measurement. Fig.3.16 presents the requisite commands in Ngspice for this analysis.

3.3 Summary

In this chapter, a Low Noise Amplifier (LNA) was meticulously designed and subsequently analyzed. A comprehensive summary of its performance metrics can be found in 3.1.

Specification	Value
Center Frequency(GHz)	2.5
$S_{11}(dB)$	$\leq -10@[2.1GHz, 2.95GHz]$
$S_{21}(dB)$	18
$Z_{11}(\Omega)$	55
$NF(dB)$	2.4
$P_{DC}(W)$	0.004
K	≥ 5
$\lvert\Delta\rvert$	< 0.8

Table 3.1: Designed LNA specifications

Chapter 4

Learning to Design Analog Circuits

The subsequent chapter is derived from our prior publication [12]. For the purposes of this thesis, the discourse will be strictly confined to the intricacies of Low Noise Amplifier (LNA) design.

4.1 Problem Statement

Human design through the use of advanced electronic design automation (EDA) tools [2] is currently the primary method for designing electronic circuits. However, human-led design is a slow process and is falling behind the human–computer co-design processes for digital circuits [22]. In order to bridge the gap and allow for faster design of analog circuits, we aim to facilitate a system that can automatically generate the parameters of an analog circuit to meet a set of performance requirements. A good system should be able to function with good accuracy across a variety of different circuit topologies.

In this work, the problem of designing one of the most important type of analog circuits, low-noise amplifiers is examined.It is noteworthy that the selected performance metrics,

themselves diverse across the various circuits, exhibit different kinds of correlations and tradeoffs.

4.1.1 Exact Specification

For a specified circuit topology, let n be the number of component parameters, such as resistances, transistor widths, and voltages. Let $X_1, \ldots, X_n$ be the operational ranges of each of these parameters, and $X = =_{i=1}^{n} X_i$ the design space. We assume the availability of a simulator $f : X \to Y$, where $Y = R_+^k$ is the positive orthant of the real vector space of k performance metrics of interest.

The problem of design from exact specification is that of finding a function $g \approx f^{-1} : Y \to X$ such that, when a user specifies target performance $y \in Y$, the system can suggest a design $\hat{x} = g(y)$. Upon suggesting $\hat{x}$, it can be simulated to measure its performance $\hat{y} = f(\hat{x})$. The *error* of the system is measured by the relative difference in its performance metrics

$$\delta_i = \frac{|y_i - \hat{y}_i|}{y_i}. \tag{4.1}$$

For evaluation, the relative error is averaged across multiple test points as well as across the k metrics. We also measure the success rate as the fraction of test points with relative error within a given margin. We note that, in a real-world system, users can input a target performance vector y for which no circuit exists with low error. The system can use the simulator to check that the predicted circuit $g(y)$ is incorrect, but it is a hard problem to determine whether another circuit would be correct, particularly if the instance is out-of distribution for the data used to train the system. We therefore focus on evaluating the system on in-distribution data $y \in f(X)$, and leave the challenging and interesting question of out of distribution generalization to future work.

4.1.2 Threshold Specification

When manual circuit design is challenging, guessing a feasible performance vector $y \in f(X)$ can be just as challenging, particularly if it consists of many metrics that are subject to intricate tradeoffs. Instead, it would be easier for a user to specify performance thresholds that the designed circuit should meet. We denote by λ_i the threshold direction of metric i, i.e. $\lambda_i = 1$ or -1 respectively whether it is majorative (the more the better) or minorative (the less the better).

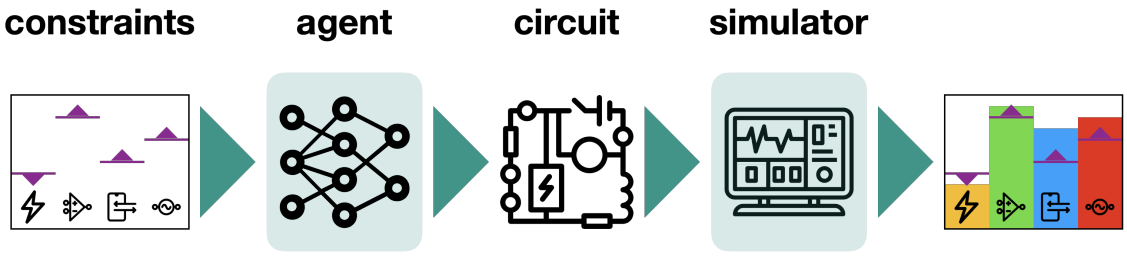

Figure 4.1: The problem of automated design by threshold specification. A user specifies threshold constraints on the circuit's performance metrics. A design agent then generates a circuit that, when simulated, meets the constraints.

The problem of design from threshold specification (fig.4.1) is that of finding a function $g : Y \to X$ such that, when a user specifies target performance thresholds $y \in Y$, the suggested design $\hat{x} = g(y)$ aims to meet the thresholds y by having its simulated performance $\hat{y} = f(\hat{x})$ satisfy $\lambda\hat{y} \geq \lambda y$ element-wise. The error of this system is measured by the relative amount of threshold violation

$$\delta_i = \frac{\max\{\lambda_i(y_i - \hat{y}_i), 0\}}{y_i}. \tag{4.2}$$

As before, we measure success rate by the fraction of test data for which the thresholds for all metrics are met up to a given error margin.

To evaluate a system solving the threshold specification problem, we should use threshold

queries that follow a similar distribution to that of real users. Leaving user studies to future work, we approximate this distribution by perturbing simulated performance metrics similarly to [15]. Given the measured performance $y = f(x)$ of a simulated circuit x, we sample standard uniform perturbations $u \sim \mathrm{U}^k$ for the k metrics, independent and identically distributed (i.i.d), and use the perturbed vector

$$\tilde{y}_i = (1 - \epsilon\lambda_i u_i) y_i \tag{4.3}$$

as the threshold query. Here ϵ is the perturbation magnitude hyperparameter; in this work we use $\epsilon = 0.2$. Note that, by construction, $\lambda y \geq \lambda \tilde{y}$, so that there always exists a circuit (namely, x) that meets the threshold $\tilde{y}$.

4.2 Analog Automated Circuit Design

Automating the design of analog circuits has been studied before, particularly in operational amplifiers (op-amps) that are specified by their voltage gain, bandwidth, and power consumption (for a survey, see [16]). [26] proposes a reinforcement learning (RL) approach to designing 3-stage amplifier circuits from threshold specification. Similarly, [23]adopts RL to design 2-stage operational amplifiers. While RL is readily amenable to threshold constraints, it suffers from poor data efficiency compared with supervised learning approaches [16]. [25] uses supervised regression to design another type of circuits, a 4-bit current-steering Digital-to-Analog converters (DAC), from exact specification of the performance metrics. Other works have used supervised learning to design various opamps [18][15][17] with varying and often incomparable data efficiency [16]. In this work, we step beyond the scope of op-amp design to additionally investigate the design of other critical analog circuit blocks, in particular radio-frequency electronic circuits that are commonly used in cellular communication applications [1]. It is noteworthy that some of the selected circuits, e.g., mixers and oscil-

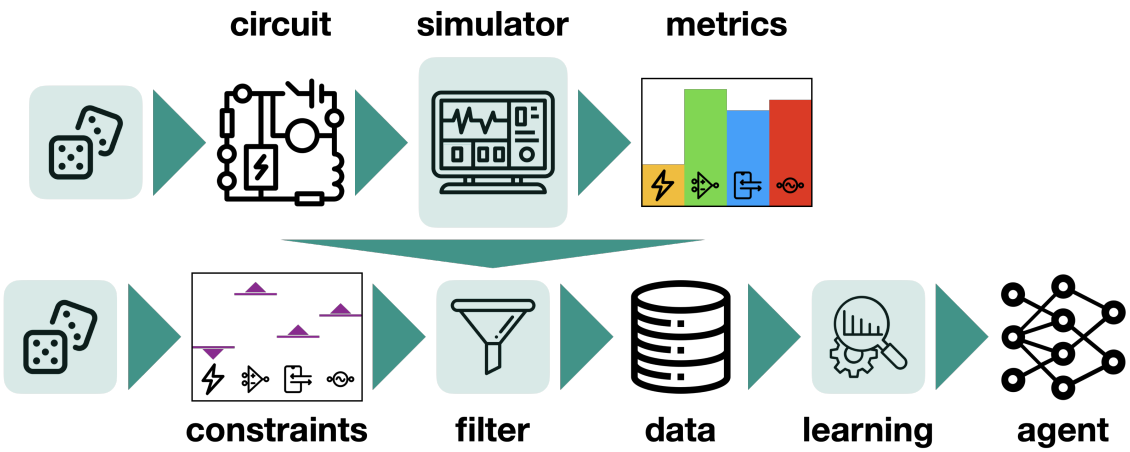

Figure 4.2: Proposed method for automated design from threshold specification. Circuit parameters are sampled within a user-defined range, simulated, and measured. Performance metrics are randomly adjusted to sample threshold queries. A data filtering process then generates training data for supervised learning a circuit design agent that generates circuits to meet threshold requirements.

lators, are among the most nonlinear analog circuits with high sensitivity to variations in design parameters. Our results further show that design agents for amplifiers as well as more intricate circuits can be learned by supervised regression from much smaller data sets than previously accomplished. Finally, we learn to design these circuits from threshold specification, in contrast to most previous supervised learning works. [15] previously considered this setting, and proposed a method that is reproduce in this work under the name D_m.

We show that this method can lead to suboptimal performance, analyze the reason through an ablation study, and propose a new method that mitigates this issue.

4.3 Method

We use supervised learning to approximate the inverse of the simulator function mapping circuit parameters to performance metrics (fig.4.2). We interface an external simulator to generate a dataset D_0 consisting of circuit parameter vectors $x \in R^n$ and their respective

measured performance metrics vectors $y \in R^k$. We (optionally) pass this dataset through a filtering pipeline that prepares it for solving the threshold specification problem Finally, we employ a supervised learning algorithm, such as gradient-based optimization, to train a design agent. In this section we describe the system components: the simulator, the agent model, and several alternatives for the filtering pipeline.

4.3.1 Simulator

In this work, we use the NgSpice simulator [19]. The circuit topology and its fixed parameters, as well as the simulation parameters, are provided to the simulator via a format called netlist [13]. In addition to the netlist, the simulator loads analysis commands that determine how it measures the performance metrics of interest. For some circuits, multiple analysis commands are given to measure the circuit under distinct conditions.

The external simulator is wrapped by a Python interface to allow easy access to two functionalities. First, to generate simulation data, a user inputs the range and step size of each circuit parameter, and the simulator loops through this grid to output a dataset D_0 of parameter–metrics pairs. Second, to evaluate the trained model, predicted circuit parameters are input to the simulator, and the measured performance is compared with the target performance.

4.3.2 Agent Model

Before the raw data from the simulator can be put through the model, we apply a few data pre-processing steps. The different features of the data have vastly different scales. In order to allow the model to learn across such different scales, we first shift and scale all values to the range $[-1, 1]$. This normalization is applied both to the performance metrics before

they are fed to the model and to the ground-truth circuit parameters used for training, and an appropriate inverse operator is applied to the model's parameter predictions. In this work, we experiment with three different agent models. The main model is a neural network with an architecture of a simple multi-layer perceptron, trained with the Adam optimizer [11]. The network takes in a vector of desired performance metrics and predicts a vector of circuit parameters, which is then compared with the ground-truth parameters using an absolute (L_1) loss. The sizes of the first and last layers of the network are adjusted to reflect the number of performance metrics and circuit parameters, respectively, which are different for each experiment described in the experiment section. An alternative model we consider is ensembles of decision trees trained with the Random Forests algorithm [5]. Finally, to assess the need for any learning at all, we compare with a lookup method that memorizes the training data and selects, for each test performance vector, the training circuit that minimizes the relative performance error.

4.3.3 Filtering Pipeline

The problem of design from exact specification can be solved by supervised learning, in which the training set is the simulation dataset D_0, inverted so that performance metrics y are inputs and circuit parameters x are outputs. However, this method is unlikely to be sufficient for the threshold specification problem , in which some threshold vectors are out-of-distribution for D_0, because no circuit has them as its exact performance. We therefore propose a filtering pipeline that constructs, from the same D_0, a second dataset which, when used for supervised learning, trains a model that predicts circuit parameters from threshold specification.

To prepare a circuit for the threshold specification problem, two properties of the metrics vector need to be provided. First, because some metrics, such as gain or bandwidth, are

majorative (the more the better), while others, such as power consumption, are minorative (the less the better), we need to know for each metric i its threshold direction $\lambda_i \in \{-1, 1\}$. Second, a specification asking for the highest gain at power consumption at most p is different from one asking for the lowest power consumption that achieves gain at least g. We may therefore have a preference order over metrics, such that we lexicographically prefer improving y_i over improving y_j, whenever $i < j$, as long as all threshold constraints are approximately met. We say that y is lexicographically better than y' if there exists i such that $y_j = y'_j$ for all $j < i$ and $\lambda_i y_i > \lambda_i y'_i$. The filtering pipeline starts by finding, for each performance vector $y \in D_0$, all *feasible* performance vectors $y' \in D_0$ that meet the threshold specification y, i.e.

$$F(y; D_0) = \{(x, y') \in D_0 | \lambda y' \geq \lambda y\}. \tag{4.4}$$

The design agent needs to map the threshold specification y to one such $x \in F(y)$, but it may not be immediately clear which one. We hypothesize that, crucially to learning with high success rate from small datasets, our training dataset must be systematic in selecting a representative of $F(y)$. This systematicity manifests as a pattern that the learning algorithm can generalize, whereas including the entire $F(y)$ or selecting from it sporadically might lead to conflicts that impede generalization. We propose to select the lexicographically best training-set circuit that meets the threshold

$$\bar{D}_0^* = \{(x, y) | y \in D_0, x = F(y; D_0)\} \tag{4.5}$$

where we select for x a single representative (x, y') of $F(y)$ that maximizes y' lexicographically. In the notation $\bar{D}_0^*$, the bar denotes feasibility of x for y and the star denotes selection of the best representative. We note that, by definition, all members of $F(y)$ have good circuit parameters that meet the threshold y. However, adding all of them to our training set, similar to the method proposed by [15], would create conflicts where the same network

input y is mapped to different outputs. By breaking "ties" in a consistent way — and in accordance with user-specified preference over metrics we create a dataset more conducive of learning. The new dataset $\bar{D}_0^*$ has the same size as the simulation dataset D_0 and the same set of performance vectors. The circuit parameter vectors in $\bar{D}_0^*$ are those that define its Pareto frontier, that is, for which no other simulated circuit is better in all performance metrics. Thus, $\bar{D}_0^*$ consistently maps feasible performance vectors to frontier circuits.

Threshold Queries

Previously we, discussed how performance metrics measured in simulation are perturbed to generate threshold queries (Eq. (4.3)). We denote thus perturbed data by

$$D_\epsilon = \{(x, (1 - \epsilon\lambda u)y) | (x, y) \in D_0, u \sim U^k \text{ i.i.d}\}. \tag{4.6}$$

Note that the distribution of threshold queries $y \sim D_\epsilon$ is different than the distribution of simulated metrics vectors $y \sim \bar{D}_0^*$. To avoid a mismatch of the training and test distributions, we combine the filters to form a dataset of threshold queries with a principled selection of target circuits:

$$\bar{D}_\epsilon^* = \{(x, \tilde{y}) | \tilde{y} \in D_\epsilon, x = F(\tilde{y}; D_0)\}. \tag{4.7}$$

$\bar{D}_\epsilon^*$ is a dataset mapping ϵ-perturbed metrics vectors $\tilde{y}$ to circuits whose (unperturbed) simulated metrics are feasible for the threshold query $\tilde{y}$, selecting the lexicographically best such circuit.

Baseline and Ablation

We compare our dataset construction methods, $\bar{D}_0^*$ and $\bar{D}_\epsilon^*$, with a baseline that closely follows [15]. We define D_ϵ^m as the union of m i.i.d. samples of D_ϵ

$$D_\epsilon^m = \bigcup_{t=1}^{m} D_\epsilon[u_t]; \quad u_t \sim U^k \text{ i.i.d.} \tag{4.8}$$

In our experiments, $m = 20$. The reasons are that by construction, in each $(x, \tilde{y}) \in D_\epsilon^m$ the circuit x is feasible for the threshold query $\tilde{y}$, i.e. $\lambda f(x) \geq \lambda \tilde{y}$; and that the training distribution $\tilde{y} \sim D_\epsilon^m$ is identical to our evaluation distribution $\tilde{y} \sim D_\epsilon$. Note that, in contrast to most of the literature on analog circuit design automation via supervised learning, which employs a simulation dataset akin to D_0, D_ϵ^m is suited for the threshold specification problem [15]. Unfortunately, the dataset D_ϵ^m can be very confusing to learn from. Because the simulator function f is not necessarily injective, there may exist multiple circuits with similar performance vectors. Moreover, such vectors have overlapping supports of their perturbation distributions. The result is that D_ϵ^m will tend to have similar threshold queries mapped to vastly different circuit parameters, rendering their prediction difficult.

We propose an ablation that more directly demonstrates this issue. In $\bar{D}_\epsilon^m$, we select for each $\tilde{y} \in D_\epsilon$ the m lexicographically best feasible circuits, rather than only the single best in $\bar{D}_\epsilon^*$ (Eq. (4.7)):

$$\bar{D}_\epsilon^m = \{(x, \tilde{y}) | \tilde{y} \in D_\epsilon, x \in F(\tilde{y}; D_0)\}. \tag{4.9}$$

We expect this method to perform suboptimally, more similarly to D_ϵ^m than to $\bar{D}_\epsilon^*$. This would provide evidence that the main aspect impacting the prior method, compared with the novel one, is the existence of multiple targets for each query, rather than the other differences namely, the selection of circuits from the feasible set $F(y)$, or the preference of

lexicographically better circuits.

To summarize, we consider six datasets: (1) D_0 is the simulation data; (2) D_ϵ has perturbed performance metrics that resemble the threshold query distribution, and is used for method evaluation; (3) $\bar{D}_0^*$ and (4) $\bar{D}_\epsilon^*$ are our proposed methods, without and with perturbation to match the test distribution; (5) D_ϵ^m is a baseline similar to [26]; and (6) $\bar{D}_\epsilon^m$ is an ablation study.

4.4 Experiments

We experiment with our methods on a diverse group of seven circuit topologies, detailed below. Best practices in circuit design suggest that circuit parameters are chosen based on their impact on performance metrics [3][4][10]. Only these parameters are used to optimize performance for each circuit. The simulated LNA Circuit is using a parameter grid consisting 4096 points, 4 parameters, and hence 1024 points per parameter.

To facilitate result reproduction. The supplementary details of the LNA circuit employed in our experiments can be found in Tables 4.1, and 4.2.

Our main method uses the $\bar{D}_\epsilon^*$ dataset to train a neural network and evaluate its success rate in 10-fold cross-validation. For each circuit topology, we perform three comparisons of this method. First, we compare the main method with the five other data construction methods described in the previous section. Second, we compare the gradient-based learning algorithm with Random Forests and a simple lookup method. Third, we study the sensitivity to the amount of training data by varying it. We compare the success rate of 10-fold cross validation, which uses 90% of the data for training each fold, with using 5%, 10%, 20% and 50% of the data for training. We do this by randomly splitting the data into (respectively) 20, 10, 5, and 2 disjoint subsets, training on one subset, testing on the rest, and then averaging

the result across the splits.

The solid curve is the average over 10 runs of data splitting and training, and the shaded area is the standard-error of the mean (SEM) over those runs.

Table 4.1: Range LNA Performance Metrics

Performance Metric	**LNA**	
	Min	Max
Power Gain (db)	5.16	18.65
S_{11}(db)	-19.1	-17.3
NF(db)	2.15	2.39

Table 4.2: LNA Design Parameters and Range of Variations

Circuit	**Variable**	Start	Step	End
LNA	$M_{1,2}$[w]	73um	0.5um	76.5um
	L_g	9.4nH	0.2nH	10.8nH
	L_s	747pH	1pH	754pH
	L_d	3.7nH	0.1nH	4.4nH

4.4.1 Performance Metric Ordering Variations

In our study, we subjected the LNA circuit to an assessment using two distinct orders of performance metrics: Order A (Power Gain, S_{11}, NF), and Order B (S_{11}, NF, Power Gain). We optimized for maximizing Power Gain and minimizing S_{11} and NF in both orders. Notably, the circuits generated by Order A showcased an average Power Gain that was larger (thus better) by 0.84 dB compared to those generated by Order B. Additionally, these circuits exhibited an average S_{11} that was higher (thus worse) by 0.53 dB in comparison, concluding that the user-specified order of performance metrics effectively creates the desired preference over them.

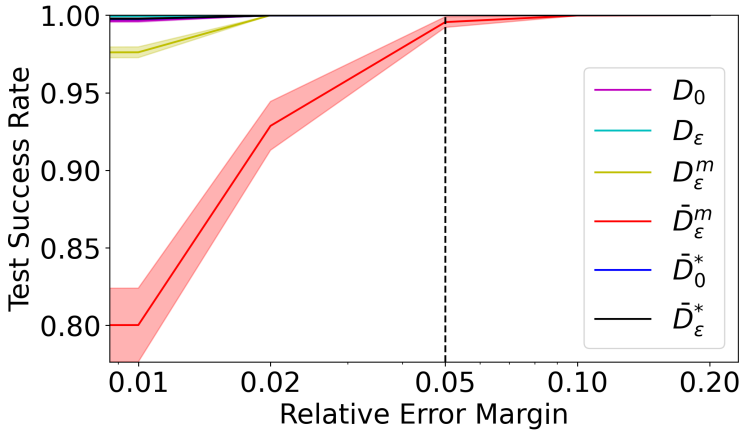

Figure 4.3: Comparing success rate in the threshold specification problem for different training datasets for the low-noise amplifier Circuit

Table 4.3: Circuit Comparison Info at 1 %

ML/Circuit	**LNA**
Lookup	0.998 ± 0.001
NN	0.998 ± 0.0
RF	0.995 ± 0.001

Table 4.4: Circuit Data Size Comparison Info at 1 %

LNA	D_O	D_ϵ^*
0.05	0.994 ± 0.005	0.972 ± 0.003
0.1	0.993 ± 0.012	0.985 ± 0.003
0.2	0.991 ± 0.003	0.991 ± 0.002
0.5	0.992 ± 0.007	0.996 ± 0.003
0.9	0.998 ± 0.001	0.997 ± 0.001

Table 4.5: Circuit Comparison Info at 1 %

D/Circuit	**LNA**
D_0	0.996 ± 0.001
D_ϵ	0.957 ± 0.009
D_ϵ^m	0.542 ± 0.003
$\bar{D}_\epsilon^m$	0.801 ± 0.024
$\bar{D}_0^*$	0.847 ± 0.021
$\bar{D}_\epsilon^*$	0.936 ± 0.012

Table 4.6: Comparing Results with the previous work

	Performance Metric	This Work (%)	Best Reported (%)	Related Works
LNA	G_T	0.0028±0.001	< 5	[7]
	S_{11}	0.007±0.001		
	NF	0.002±0.0005		

Bibliography

[1] *RF microelectronics*, volume 2. 2012.

[2] E. Afacan, N. C. C. Lourenço, R. M. F. Martins, and G. Dündar. Review: Machine learning techniques in analog/rf integrated circuit design, synthesis, layout, and test. *Integr.*, 77:113–130, 2021.

[3] J. Bandler and S. Chen. Circuit optimization: the state of the art. *IEEE Transactions on Microwave Theory and Techniques*, 36(2):424–443, 1988.

[4] J. W. Bandler and J. E. Rayas-Sánchez. An early history of optimization technology for automated design of microwave circuits. *IEEE Journal of Microwaves*, 3(1):319–337, 2023.

[5] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, oct 2001.

[6] R. E. Collin. *SolidState Amplifiers*, pages 713–798. 2001.

[7] E. Dumesnil, F. Nabki, and M. Boukadoum. Rf-lna circuit synthesis by genetic algorithm-specified artificial neural network. *2014 21st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 758–761, 2014.

[8] R. H. Friss. Noise figure of radio receivers. *Proceedings of the Institute of Radio Engineers*, 32(9):679–692, 1944.

[9] G. Gonzalez. *Microwave transistor amplifiers: Analysis and Design*. Prentice Hall, 1997.

[10] A.-K. S. O. Hassan, A. S. Mohamed, A. A. Rabie, and A. S. Etman. A novel surrogate-based approach for optimal design of electromagnetic-based circuits. *Engineering Optimization*, 48:185 – 198, 2016.

[11] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[12] D. Krylov, P. Khajeh, J. Ouyang, T. Reeves, T. Liu, H. Ajmal, H. Aghasi, and R. Fox. Learning to design analog circuits to meet threshold specifications, 2023. URL:`https://doi.org/10.48550/arXiv.2307.13861`.

[13] F. Lannutti, P. Nenzi, and M. Olivieri. Klu sparse direct linear solver implementation into ngspice. In *Proceedings of the 19th International Conference Mixed Design of Integrated Circuits and Systems - MIXDES 2012*, pages 69–73, 2012.

[14] T. H. Lee. *The Design of CMOS Radio-Frequency Integrated Circuits*. Cambridge University Press, 1998.

[15] N. Lourenço, J. Rosa, R. Martins, H. Aidos, A. Canelas, R. Povoa, and N. Horta. On the exploration of promising analog ic designs via artificial neural networks. pages 133–136, 07 2018.

[16] R. Mina, C. Jabbour, and G. E. Sakr. A review of machine learning techniques in analog integrated circuit design automation. *Electronics*, 11(3), 2022.

[17] S. D. Murphy and K. G. McCarthy. Automated design of cmos operational amplifier using a neural network. *2021 32nd Irish Signals and Systems Conference (ISSC)*, pages 1–6, 2021.

[18] H. M.V. and B. P. Harish. Artificial neural network model for design optimization of 2-stage op-amp. In *2020 24th International Symposium on VLSI Design and Test (VDAT)*, pages 1–5, 2020.

[19] Ngspice Development Team. *Ngspice User's Manual*. Ngspice Project, 2023. Accessed: 2023-08-15.

[20] D. M. Pozar. *Microwave engineering*. John wiley & sons, 2011.

[21] B. Razavi. Rf ic design challenges. In *Proceedings of the 35th Annual Design Automation Conference*, DAC '98, page 408–413, New York, NY, USA, 1998. Association for Computing Machinery.

[22] G. Renner and A. Ekárt. Genetic algorithms in computer aided design. *Computer-Aided Design*, 35(8):709–726, 2003. Genetic Algorithms.

[23] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolić. Autockt: Deep reinforcement learning of analog circuit designs. *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 490–495, 2020.

[24] D. Shaeffer and T. Lee. A 1.5-v, 1.5-ghz cmos low noise amplifier. *IEEE Journal of Solid-State Circuits*, 32(5):745–759, 1997.

[25] R. A. Vural, N. Kahraman, B. Erkmen, and T. Yildirim. Process independent automated sizing methodology for current steering dac. *International Journal of Electronics*, 102:1713 – 1734, 2015.

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

[27] P. Yip. *High-frequency circuit design and measurements*. Springer Science & Business Media, 2012.