Speeding-Up Complex RF IC Sizing Optimizations with a Process, Voltage and Temperature Corner Performance Estimator using Deep ANNs

Pedro José da Costa Diogo Caldinhas Vaz

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor: Professor Nuno Cavaco Gomes Horta

Doctor Ricardo Miguel Ferreira Martins

Examination Committee

Chairperson: Professor Teresa Maria Sá Ferreira Vazão Vasques

Supervisor: Professor Nuno Cavaco Gomes Horta

Member of the Committee: Professor Manuel Fernando Martins de Barros

November 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfils all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would like to acknowledge my supervisors Prof. Nuno Cavaco Gomes Horta and Prof. Ricardo Miguel Ferreira Martins, for all their guidance and support throughout the development of this dissertation and a special thanks to Prof. Nuno Lourenço for all his availability and advice throughout the experimentations performed in this work which, without his help, would not have accomplished any satisfying results.

I would like to say thank you to Instituto de Telecomunicações for hosting me in its facilities. This research institution which is funded by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/50008/2020.

I would also like to thank my parents and brothers for their friendship and encouragement throughout my entire academic life. Without them, I would certainly not be the person I am today.

Finally, to all my colleagues and friends that accompanied me in this journey and helped me grow as person. Thank you.

Abstract

The work presented in this dissertation belongs to the scientific area of electronic design automation and addresses the automatic sizing of radio-frequency (RF) integrated circuits (ICs).

This work explores an innovative approach to automatic circuit sizing of RF IC blocks using deep learning techniques and, more specifically, artificial neural networks (ANNs), to complement and improve an optimization-based sizing loop. In order to find all the relevant performance figures of a certain circuit sizing solution, the optimization loop simulates and evaluates the desired circuit topology under different process fabrication dispersions, as well as voltage and temperature variations, which are also known as process, voltage, and temperature (PVT) corner analysis.

The ANN architecture proposed in this work is a regression-only model. The goal of this model is to estimate all the relevant circuit performances in PVT corners using, as input features, the circuit's sizing and the accurate performance figures in typical conditions, and thus, speeding-up the optimization process by bypassing the time-consuming circuit simulation. This model will complement and speed-up the optimization loop of the AIDA tool.

The results obtained show that the PVT estimator was able to reduce the workload of the circuit simulator up to 78.5% while achieving a total optimization speed-up factor of 2.92. These results proved the PVT estimator's plug-and-play capabilities, being reused for optimizations with completely different targets of the same circuit topology, while its structure can be additionally reused for different VCO circuit topologies.

Keywords

Artificial Neural Networks, Automatic Sizing Optimization, Circuit Sizing, Deep Learning, Electronic Design Automation, Radio-Frequency Integrated Circuits

Resumo

O trabalho apresentado nesta dissertação pertence à área científica de automação de projeto eletrónico e aborda o dimensionamento automático de circuitos integrados de rádio frequência.

Este trabalho explora uma abordagem inovadora para dimensionamento automático de blocos de circuitos integrados de rádio frequência usando técnicas de aprendizagem profunda e, mais especificamente, redes neuronais artificiais para complementar e melhorar um ciclo de dimensionamento baseado em otimização. Para encontrar as performances para uma certa solução de dimensionamento, o ciclo de otimização simula o circuito em diferentes dispersões de processamento de fabrico, e ainda variações de tensão e temperatura, mais conhecido como análise de condições extremas de funcionamento PVT.

A arquitetura da rede neuronal artificial proposta neste trabalho é a de um modelo de regressão. O objetivo do modelo é estimar os desempenhos do circuito mais relevantes nas condições extremas de funcionamento PVT usando, como entradas do modelo, o dimensionamento do circuito e as performances em condições típicas, e assim, acelerar o processo de otimização evitando simulações demoradas. Este modelo vai complementar e acelerar o ciclo de otimização da ferramenta AIDA.

Os resultados obtidos mostram que o estimador de PVT foi capaz de reduzir o esforço computacional do simulador de circuitos até 78.5%, alcançando um fator de 2.92 de aceleração total do tempo de otimização. Estes resultados também mostram que o estimador de PVT pode ser reutilizado para otimizações com objetivos bastante distintos para a mesma topologia de circuito, enquanto que a sua estrutura pode ser reutilizada para diferentes topologias de circuitos VCO.

Palavras-chave

Aprendizagem Profunda, Automação de Projeto Eletrónico, Circuitos Integrados de Rádio Frequência, Dimensionamento de Circuitos, Otimização Automática de Dimensionamento, Redes Neuronais Artificiais

Table of contents

List of Figures

List of Tables

Acronyms

5G – 5th Generation Broadband

AMS – Analog or Mixed-Signal

ANN – Artificial Neural Network

CAD – Computer-aided Design

EDA – Electronic Design Automation

ELU – Exponential Linear Unit

IC – Integrated Circuit

IoT – Internet of Things

ML – Machine Learning

MOO – Multi-Objective Optimization

MS – Mixed-Signal

MAE – Mean Absolute Error

MAPE – Mean Absolute Percentage Error

MSE – Mean Squared Error

OTA – Operational Transconductance Amplifier

PCA – Principal Component Analysis

POF – Pareto Optimal Front

PVT – Process, Voltage and Temperature

ReLU – Rectified Linear Unit

RF – Radio Frequency

RL – Reinforcement Learning

SoC – Systems-on-Chip

SOO – Single-objective Optimization

SVM – Support Vector Machine

VCO – Voltage-controlled Oscillator

1 Introduction

Over the last few years, the electronics industry has experienced a massive increase in the demand for smaller and more complex integrated systems, mainly due to the rise of portable devices. Now more than ever, developers are faced with the challenge of creating more powerful systems while ensuring smaller size and low power consumption. Technologies such as the internet of things (IoT) or 5th generation broadband (5G), will join millions of devices and sensors together, enabling great advances in education, healthcare, transportation, agriculture, amongst many other areas. All these applications continuously gather an increasing amount of data, posing unprecedented challenges to each element of the networks. Due to this, today's market demands high communication rates, large bandwidths, and ultralow-power consumptions, in which radio frequency (RF) integrated circuits (ICs) play a critical role. Most of these systems implement a combination of both analog/RF and digital circuits in the same chip, i.e., a Mixed-Signal (MS) Systems-on-Chip (SoC).

1.1 Motivation: The Mixed-Signal Design Paradigm

Although the percentage of area occupied in MS SoCs by digital circuits is larger than the area occupied by analog circuits, the analog parts generally require a higher effort from the circuit designer, as depicted in Fig. 1.1.

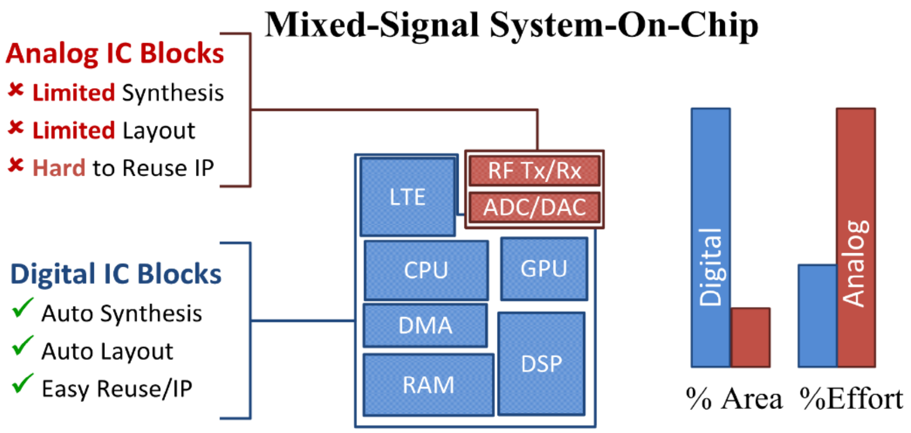


Figure 1.1 - Contrast between Analog and Digital blocks' area of implementation in an IC and the corresponding effort to implement them from the perspective of a designer. Reprinted from [1]

The continuous nature of the signal values handled by analog circuits makes them more susceptible to noise and process variations than digital circuits, resulting in a design that is both more complex and time demanding.

Because of this, designers throughout the years have been trying to replace analog circuits with digital ones. However, some typical blocks cannot be replaced. A few examples are shown in the following list [2]:

- In a system's input, the signals captured by a sensor, microphone, or antenna must be received, amplified, and filtered to allow their digitalization with a good signal-to-noise and distortion ratio. Common applications of these circuits are in sensor interfaces, telecommunication receivers or sound recording;
- In a system's output, the processed digital signal must be converted to analog, and to drive outside load with low distortion, it must be strengthened. These circuits are typically utilized in telecommunication transmitters and speakers;
- MS circuits like sample-and-hold, analog-to-digital converters, and frequency synthesizers. These blocks establish the interface between input/output sides of a system and digital processing parts of a SoC;
- Voltage or current reference circuits and crystal oscillators to offer stable and absolute references for the circuits mentioned before.

In addition, for IoT and 5G, the design of RF and mm-Wave ICs in deep nanometric technologies is becoming extremely difficult because of their high complexity, demanding performances, and their need to be designed and manufactured at minimal costs under strict time-to-market constraints.

The design problems faced are not only because of the overwhelming wide range of frequencies and dynamic ranges involved but also due to their dependence on non-reliable models of passive devices; the huge negative impact of layout parasitics at high frequencies; and being integrated on deep nanometer technologies suffering from unprecedented variability problems and non-idealities. Avoiding costly redesign cycles and reducing post-fabrication tuning and compensation work on first-pass fabrication success became primary RF IC design objectives. Established computer-aided design (CAD) companies provide environments that allow circuit designers to carry this flow manually. Despite this, the classical trial and error method is no longer viable due to the high number of complex interactions leading to sub-optimal RF designs. To solve these issues and comply with the requirements above, electronic design automation (EDA) tools able to explore design spaces beyond human capabilities would greatly benefit designers. The adoption of automation mechanisms can greatly reduce circuit development time while simultaneously improving their performance. Unlike analog/RF IC design, plenty of EDA tools are available and established for the digital IC design flow. For these reasons, the development and improvement of automation tools for the analog design process became an extremely relevant topic of research.

Despite the effort of the research community on this topic over the past few decades, automatic analog IC design tools have not yet reached the desirable state of maturity, resulting in a continuous human intervention throughout the whole design process.

To summarize, the process of designing an analog block generally requires more effort than a digital block, and the designer must be knowledgeable and skillful. Adapting and improving existing EDA tools will result in the computational support that analog/RF designers require to face the abovementioned challenges.

1.2 Topic Overview: Analog/RF IC Design Flow

Regarding the specific design flow of analog ICs, each designer/company may have its own IC design flow. However, in [2], Gielen and Rutenbar standardized the steps that most designers take when manually designing an analog or mixed-signal (AMS) IC, introducing the well-accepted design flow shown in Fig 1.2. This flow consists of a series of top-down design steps repeated from the system-level to the device-level and bottom-up layout generation and verification.

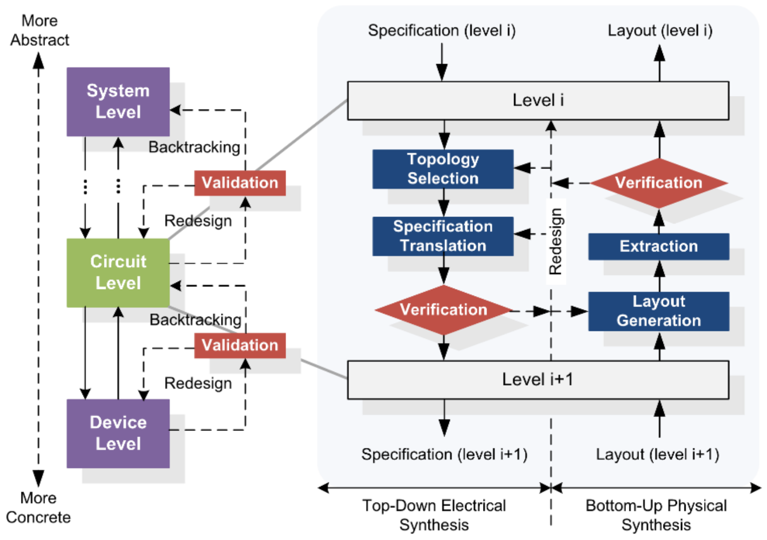


Figure 1.2 - Hierarchical levels and design tasks of AMS IC design process. Reprinted from [3].

Using a hierarchical top-down design methodology allows performing system architectural exploration, achieving a better overall system optimization at a higher level of abstraction before starting more specific implementations at the circuit or device levels. With this, one can find problems earlier in the AMS design flow, increasing the chances of first-time success, with fewer time-consuming redesign iterations [4].

In this design flow, the number of hierarchy levels may vary according to the complexity of the system being designed, and although there are no overall accepted representations for the architectural design, the steps between two hierarchical levels are:

- Top-down electrical synthesis path, which includes topology selection, specification translation (or circuit sizing at the lowest level), and design verification;
- Bottom-up physical synthesis path, that includes layout generation and detailed design verification (after layout extraction).

In topology selection, the most appropriate circuit topology is determined to meet a certain number of specifications at the current hierarchy level. This topology can be chosen from a set of existing topologies or synthesized.

Specification translation is the step where the designer maps the high-level block specifications and, given a certain topology, maps them into individual specifications for each of the sub-blocks. At the lowest level, due to these sub-blocks being single devices, this task is narrowed down to circuit sizing. Before proceeding down in the hierarchy, specifications translation is verified using simulation. At higher levels of the design flow, there is no device-level sizing available, which results in behavioral simulations. However, at lower levels (circuit and device-level), device sizing is available, and therefore, electrical simulations are used. Each block specifications are passed to the next hierarchy level, and the whole process is repeated until the top-down electrical synthesis flow is completed.

To aid designers overcome the many difficulties encountered in manual sizing of analog/RF IC blocks, several optimization-based sizing approaches emerged. These EDA tools use several algorithms that explore the design space effectively, rather than iterating over designer-defined analytical equations. They can be used along with performance models that can capture several circuit characteristics of RF circuits. However, despite its increased computational effort, utilizing foundry-provided device models and a circuit simulator as an evaluation engine resulted in the most accurate and generally adopted approach. Most of the commercially available solutions that use the simulation-based architecture, e.g., Cadence's Virtuoso GXL [5] or MunEDA's DNO/GNO [6], still take a restrictive single-objective approach being used to semi-automate the manual sizing design process. Consequently, simulation-based techniques are a continuous subject of research of the community to face the most recent design challenges. This will be the topic of research in this dissertation.

After the top-down flow is completed at a certain level, the sizing obtained must be verified by generating the corresponding layout and testing its performance. If these prove to be satisfactory results, the design flow is finished. If not, a redesign is needed, repeating the previous steps or the whole flow again.

1.3 Research Objectives

When analyzing the state-of-the-art sizing tools available, as presented later in Chapter 2, optimization-based sizing proved to be the best tool in handling a vast number of different circuit complexities while achieving exceptional results.

However, there are some problems that must be mandatorily addressed. The optimization-based sizing loop requires a considerable time to complete. For example, in [7], the optimization-based sizing loop of a class-C/D voltage-controlled oscillator (VCO), performed with a population of 512 elements and 200 generations, took 50 hours to complete, and in [8], with a population of 256 elements and just 100 generations the optimization-based sizing loop of a class-B/C VCO, took 367 hours to complete. These optimizations were carried in a machine with an Intel-Xeon E5-2630-v3@2.40 GHz with 64 GB of RAM using 8 cores for parallel evaluation. For a developer that needs to meet the stringent time-to-market constraints, such optimizations' durations may not be viable, and thus, a speed-up is necessary.

In an optimization loop, the candidate sizing solution's relevant performances are found by simulating and evaluating the circuit under process fabrication dispersion and voltage and temperature variations, also known as PVT corner analysis, which take most of the computational effort. Therefore, the objectives of the work presented in this dissertation are the following:

- Speed-up the analog/RF IC optimization-based sizing loop of the AIDA tool [9] (further explained in Section 2.2.2) using artificial neural networks (ANNs) to complement the simulation process, and therefore reducing the simulator workload;
- The model to be developed aims at estimating circuit performance in the PVT corners using the circuit's sizing and performance on typical conditions as input features;
- Study if the model used for a particular circuit topology can be reused for optimizations with completely different targets of the same circuit topology (plug-and-play functionalities);
- Analyze if the structure of the model used for a particular VCO circuit topology can be reused for different VCO circuit topologies (plug-and-train functionalities);

To test these objectives, the proposed PVT estimator will be integrated on the AIDA tool and tested on three different circuit sizing optimizations, as detailed later in Chapter 5: a class C/D VCO for 3.5-to-4.8 GHz and 2.3-to-2.5 GHz ranges, and finally, an ultralow power class B/C VCO.

1.4. Document Structure

This document is organized as follows:

- Chapter 2 presents fundamental concepts and related work in the automation of IC sizing;
- Chapter 3 presents and describes the proposed solution to speed up an optimization loop as well as an overview of the ANN structure and tuning phase;
- Chapter 4 describes the tuning phase of the PVT estimator and presents the results of the PVT estimator before integration in the AIDA tool;
- Chapter 5 describes the AIDA integration of the PVT estimator and presents the results obtained from three different optimizations;
- Chapter 6 presents the conclusion drawn from this work and outlines future possible developments to further optimize the performance and abstraction of the PVT estimator.

2 Background and Related Work

In this chapter, the main techniques used in analog circuit sizing are analyzed and discussed. Afterwards, related work using machine learning (ML) techniques to enhance simulation-based optimizations will be presented and discussed. Finally, a brief comparison between the best techniques for this work will be presented and the most appropriate will be chosen.

The main techniques used in analog circuit sizing automation for the last few decades can be divided in two categories: knowledge-based and optimization-based. Both sizing categories are depicted in Fig. 2.1.

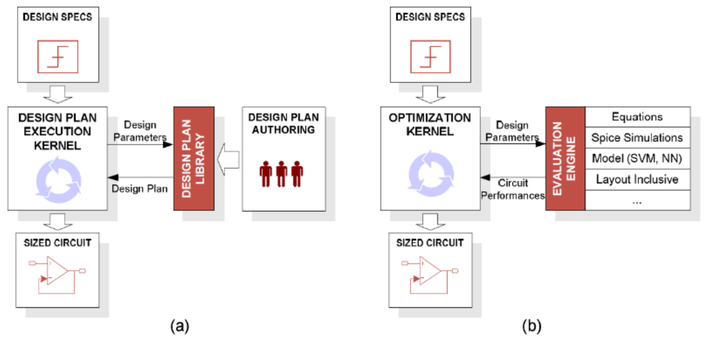


Figure 2.1 - Automatic circuit sizing methods: (a) knowledge-based and (b) optimization-based. Reprinted from [10].

2.1 Knowledge-based sizing

Regarding knowledge-based, tools such as IDAC [11] and BLADES [12] have attempted to systematize circuit design using a design plan obtained from expert knowledge. Using design equations and a design strategy, these tools generate a pre-designed plan that aims to obtain circuit component sizes that meet the performance requirements established by the designer. Despite the good results that this approach has achieved, being its short execution time the main advantage, the design plan generation process is elaborate and requires a lot of time to complete. With the fast rate of technological developments these tools require constant supervision to keep the design plan up to date and the results obtained not being optimal makes this technique only acceptable as a first-cut design.

2.2 Optimization-based sizing

Considering the strengths of knowledge-based and trying to improve its weaknesses, the next generation of sizing tools focused on the application of optimization techniques to analog/RF IC sizing. These can be categorized in two main categories regarding the method used to evaluate the circuit's performance: equation-based and simulation-based.

2.2.1 Equation-based evaluation

Equation-based optimization approaches use analytic design equations to evaluate the circuit performance. In OPASYN [13] and later in CADICS [14] design equations still had to be done by hand, while the degrees of freedom were resolved implicitly by optimization. Later, with the symbolic simulator ISAAC [15], the (simplified) design equations were automatically generated.

Due to their short evaluation time, these methods are, like knowledge-based sizing, only adequate for first cut-designs. The main problems are some design characteristics are difficult to be mapped by analytic equations and all the approximations introduced throughout the equations yield designs with low accuracy.

2.2.2 Simulation-based evaluation

These sizing methods use a circuit simulator to evaluate the circuit's performance. The main advantages of this approach over the previous ones, is its generality and easy-and-accurate model, however, due to the long execution of SPICE-based circuit synthesis and several simulation executions, this approach becomes time consuming.

One example of this approach was used in [9], where an analog IC design automation environment called AIDA implements a design flow from a circuit-level specification to a physical layout description. AIDA is a combination and integration of two in-house tools, GENOM-POF [16] and LAYGEN II [17], where the circuit synthesis is executed by GENOM-POF. AIDA's architecture is shown in Fig. 2.2.

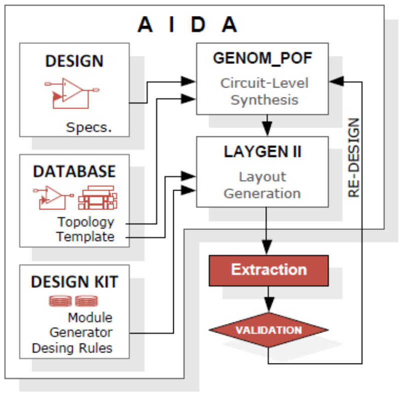


Figure 2.2 – AIDA's architecture. Reprinted from [9].

This tool uses a multi-objective design methodology together with optimization-based sizing, where a corner analysis combined with an electrical simulator as the evaluation engine, to perform fully automated circuit-level synthesis. AIDA environment was validated using a single-ended folded-cascode amplifier as a case study, where the results obtained were compared with state-of-the-art industrial simulators and analysis tools, such as Synopsys HSPICE [18] and Mentor Graphics' CALIBRE [19].

GENOM-POF is based on the multi-objective evolutionary optimization kernel NSGA-II [20] and uses the industry standard circuit simulator HSPICE. The architecture is shown in Fig. 2.3. The designer's inputs are the circuit and its testbench in the form of HSPICE netlist(s). These netlists must have, as parameters, the optimization variables and must include a method to measure the circuit's performance. The designer also must define the desired range of the optimization variables, design constraints and optimization objectives.

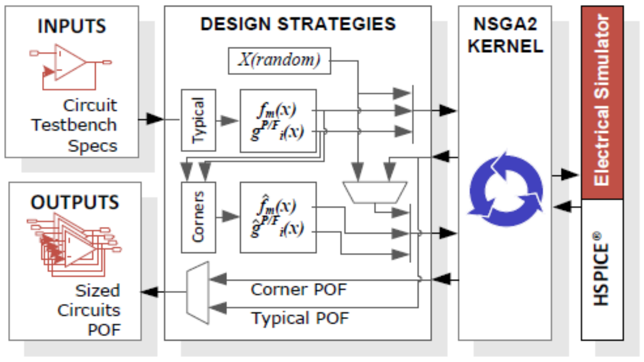


Figure 2.3 - GENOM-POF architecture based on the NSGA-II and using HSPICE simulator. Reprinted from [9].

AIDA and other similar sizing methods follow the same general flow, as its enhanced version in [8], which can be seen in Fig. 2.4. This architecture represents P candidate circuit sizing solutions proposed by the optimization engine, each representing a possible combination of design variables. At each loop iteration, the framework simulates the K test benches for each sizing of P, extracting the corresponding circuit performances. For the simulation, one can choose from several output standard formats, more specifically, .MDL, .AEX or .MEAS, where each one has a corresponding simulator, Cadence's Spectre [21], Mentor Graphics' Eldo [22] or Synopsys' HSPICE [18], respectively.

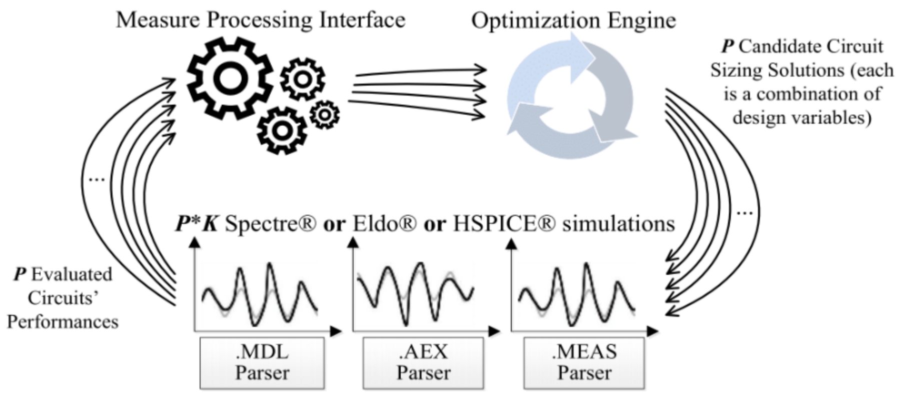

Measure Processing Interface
Optimization Engine
P Candidate Circuit Sizing Solutions (each is a combination of design variables)
...
P*K Spectre® or Eldo® or HSPICE® simulations
P Evaluated Circuits' Performances
...
.MDL Parser
.AEX Parser
.MEAS Parser

Figure 2.4 - Design flow of a multi-test bench analog and RF IC sizing optimization. Reprinted from [8].

2.3 ML in simulation-based evaluation

ML is a subset of Artificial Intelligence even though the latter aims at building complex systems and the former focus on the statistical properties of data [23]. Thomas Bayes proposed several theorems of probability theory on his essay on Probability Theory [24] that laid the theoretical foundations for statistical learning and is the cornerstone for some early ML techniques, such as Naive Bayes or Markov Chains. In 1951, the first artificial neural machine was proposed, but ANNs only began to receive more attention from the community with Frank Rosenblatt's perceptron [25] and back-propagation [26], in 1958 and 1986 respectively, where in the latter principles of dynamic programming were introduced. In the meantime, many other accomplishments have been achieved, and today there are a vast number of different ML techniques for solving classification and regression tasks.

In a classification problem, the main goal is to correctly categorize data. A common example is a simple email spam filter which assigns incoming emails to the "spam" or "not-spam" categories.

In a regression problem, the main goal of the system is to describe one or more continuous-value dependent variables as functions of the data observations. An example of regression is the prediction of house prices given the features of the house like size, number of rooms, location, etc.

A visual comparison between these two problems can be seen in Fig. 2.5.

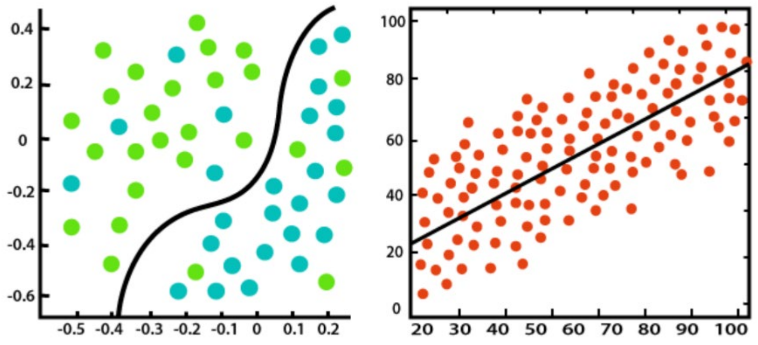


Figure 2.5 - Classification (left) vs Regression (right) problems. Picture taken from [28].

A critical characteristic of all ML systems is their ability to adapt properly to new, previously unseen data and avoid overfitting to the training data. Overfitting occurs when the system learns the detail and noise in the training data to the extent that it negatively impacts the performance of the system, affecting the system ability to generalize to new data [23][27].

Another critical attribute of a ML system is the amount and type of supervision. The following subsections refer to the different supervision categories a ML system can be categorized in.

Supervised Learning

In supervised learning, the training data of the system includes its true solution, called a label. This label can be categorical, for classification problems, or a continuous value, for regression problems. A few examples of supervised learning algorithms are linear regression, logistic regression, decision trees, support vector machines (SVM) and ANNs, among others.

Unsupervised Learning

In unsupervised learning, the training data is unlabeled, and the goal of the system is to group data points based on their features. Examples of unsupervised learning algorithms are the k-means and principal component analysis (PCA).

Semi-supervised learning

Semi-supervised learning is the third category that falls between supervised learning and unsupervised learning, in which its training data combines a small amount of labeled data with a large amount of unlabeled data and the system is trained with a combination of supervised and unsupervised algorithms. Some algorithm examples are autoencoders and deep belief networks.

With the recent technology advancement, the use of macro models like, ANNs or SVMs, introduced another type of optimization-based sizing named numerical-model based. Given the models prediction speed, using these tools, one can reduce the high simulation times of the loop simulator by aiding the simulator in certain tasks or, in a more drastic way, replacing the whole simulator.

2.3.1 Speeding-up Simulation-based Sizing with ANNs

Today, ANNs are quite popular in the ML world due to the increased amount of data and computing power available. These two factors prevented researchers from using them altogether in academic settings. Now with faster computer-processing, ANNs can be found in image processing, speech recognition and other areas where large amounts of data are available.

They are systems based in the human brain, copying the ways we learn and make decisions. These networks are composed by an input and an output layer, as well as one or more hidden layers. Each one of these layers are a combination of neurons, where the input layer is where the data is fed to the network and the output layer is where the results of the algorithm are obtained. The advantages of using ANNs are their high-performance, capability of solving problems impossible for humans, excellent algorithm for regression and classification problems and capability of handling large amounts of data. Some disadvantages are its black-box nature, i.e., difficult for researchers to completely understand why the algorithm is behaving a certain way because of how the numerical values are produced, long time to train the model and the large amount of data required.

Nonetheless, ANNs can build effective end-to-end ML systems and can be used in EDA for modeling [29], synthesis [30], layout generation [31] or even fault testing [32].

In [33], a neural network-based methodology is used to estimate the performance parameters of CMOS Operational Amplifiers (Amp-Op) topologies. To obtain the efficiency and accuracy of the resulting performance models, these were used in a genetic algorithm-based circuit synthesis system. The performance parameters of the synthesized circuits were validated by SPICE simulations and compared with the ones predicted by ANN models. Training data of the model was directly generated through SPICE simulations to provide accurate and reliable data to the system. The ANN's architecture had only one hidden layer with its number of neurons ranging from 8 to 13 to obtain good generalization and accuracy on both training and validation steps. However, in some performances predicted by the model, the test error reaches 60%.

This approach proved to be much faster compared to the traditional SPICE simulation. The genetic algorithm, using the ANN models, was executed 10 000 times for each of 8 performance parameter constraint configurations. Through SPICE simulation, each of these iterations would require 2 seconds to complete, which would be a total of about 44 hours for all configurations. The execution time using ANN models was about 10 sec for each configuration, totaling 80 sec for all configurations, which represents a speed-up factor of 2000. The models also proved to be capable of capturing nonlinear behavior of the performance characteristics of a circuit which requires a large number of simulations, but in the end the effort is justified when considering the reusability of the models in other Amp-Op topologies.

In [34], an ANN with two hidden layers is used to replace a SPICE simulator. Multi-objective optimization (MOO) is frequently used in analog sizing to reveal the trade-offs of the design specifications with the help of Pareto optimal fronts (POF). A rough POF can be found in a reasonable time with MOO, but a high-quality one requires a lot of simulator iterations which results in long synthesis times. In this paper, a method is presented to speed-up this process. After a MOO phase to obtain a low-quality POF, the process switches to a faster single-objective optimization (SOO) to complete the POF making it smoother and more continuous. At this phase the SPICE simulator was also replaced by an ANN which reduced the synthesis time even further. The training data for the ANN was the data obtained in the MOO phase.

To validate this tool, this method was applied to the design of two circuits, a Two-Stage Amplifier and a Folded Cascode Operational Transconductance Amplifier (OTA). For the first circuit a speed-up factor of about 29.7 was obtained, which translates to a 96.6% time reduction, with a maximum error of 0.44%. As for the second circuit a speed-up factor of 28.3 was obtained, which represents a 96.4% time reduction, with a maximum error of 1.55%. All the results obtained can be seen in Table 2.1.

In [35], a similar method is used to accelerate a simulation-based circuit synthesizer through the use of ANNs to determine circuit performances instead of a SPICE simulator. Instead of training the ANN with simulation data beforehand and simply replacing the simulator with the trained ANN, the simulation-based synthesizer is left unchanged for some generations of the optimization loop and only after the ANN replaces the SPICE simulator. Unlike other conventional algorithms, all the data generated in the first phase is used as training data for the ANN instead of being discarded. The proposed circuit synthesizer structure is shown in Fig. 2.6.

Table 2.1 - Results obtained for the two-stage amplifier and the folded cascode OTA

Circuit	Time			Accuracy				
	t_{SPICE} (min)	t_{nets} (min)	Improvement (%)		Gain	Bandwidth	Phase Margin	Power
Two-stage amplifier	800.70	27.07	96.62	μ_{error} (%)	0.00466	0.00502	0.00721	0.04560
				σ_{error} (%)	0.01064	0.01064	0.06115	0.10109
Folded cascode OTA	646.71	23.35	96.39	μ_{error} (%)	0.01362	0.00437	0.08254	0.00716
				σ_{error} (%)	0.10612	0.04452	0.18208	0.05252

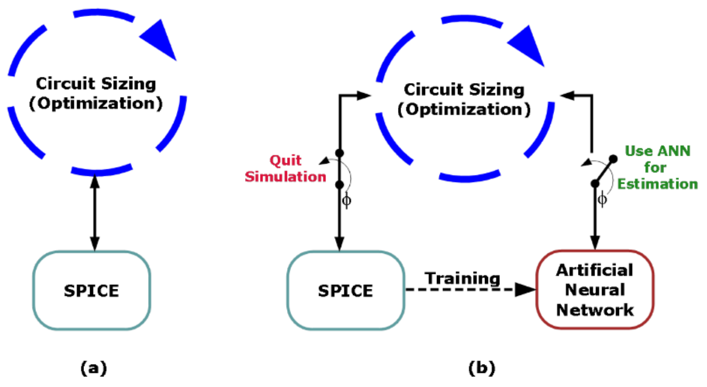

Circuit Sizing (Optimization)
SPICE
(a)
Circuit Sizing (Optimization)
Quit Simulation
ϕ
Use ANN for Estimation
ϕ
SPICE
Training
Artificial Neural Network
(b)

Figure 2.6 - a) general flow of simulation-based synthesizer b) modified flow with ANN. Reprinted from [35].

This training data acquisition step can take a lot of time and is necessary for every new topology which is one of the aspects this paper tries to solve. The main innovation of this approach is that there is no separate data acquisition step to train the ANNs used therefore makes it possible be used for every new topology without loss of generality for all analog circuits.

To validate this tool, this method was applied to the circuit synthesis phase of two circuits, a single stage amplifier and a Folded Cascode OTA. With only the SPICE simulator the circuit sizing of the single stage amplifier took 4.92 hours to complete where the optimizer iterated 100 generations. With the use of ANNs replacing the simulator after the first 20 generations, the execution time reduced by 64.8%, corresponding to a speed factor of 2.84, with errors below 1%. For the Folded Cascode OTA, which presents a higher complexity, the original optimizer took 400 generations and 22.58 hours to complete the circuit sizing. The best time reduction obtained with ANNs was 50.3% with errors below 1%, where the ANN replaced the simulator after 155 generations. All the results obtained can be seen in Table 2.2.

Table 2.2 - Execution times obtained for the single stage amplifier and folded cascode OTA

Circuit	Network usage	Time (h)	Improvement
Single stage amplifier	0%	4.92	0%
	60%	2.45	50.2%
	70%	1.99	59.6%
	80%	1.73	64.8%
Folded cascode OTA	0%	22.58	0%
	50%	12.80	43.3%
	61.25%	11.22	50.3%

In [36] ANNs are used to improve the sample efficiency for several large circuits regarding their post-layout performance parameters. ANNs are used as an oracle, where, given two different circuit sizing solutions, the ANN will predict which design performs better for each individual parameter, requiring a sub-ANN for each parameter. This discriminator achieves at least two orders of magnitude in sample efficiency which represents a big reduction in number of simulations required.

All these approaches to reduce the execution time of optimization-based sizing use ANNs to replace or complement the circuit simulator. The execution time is greatly reduced by avoiding time-consuming circuit simulations, however, to recover the accuracy lost, in [33][37] at later stages of the optimization the circuit simulator is reestablished. Furthermore, the ANN models are trained over the entire design space, which spends valuable resources modelling and evaluating large regions of unusable design combinations. In [38] the ANNs were also trained to replace the simulator, but the previous issue is somewhat addressed by applying data mining techniques to build a model that capture only significant regions of the performance space visited during automatic synthesis.

2.4 Other works of ML and Analog/RF sizing

2.4.1 Predicting sizing from performances

Instead of using ANNs to find circuit performances given a certain sizing, in recent years, the opposite was also proposed. The use of ANNs to find a circuit sizing in analog ICs is getting rather popular nowadays, and they have shown some promising results when learning to predict circuit sizing when asked for some target performance [39][43]. In [40] an ANN is used to predict device sizes for different current sources and a simple differential amplifier on different technologies. In [41] the sizing of a low noise amplifier given its target performances is performed using various sequential ANNs. This method showed good prediction accuracy, but the sequential model revealed to be difficult to tune and train. Training the model required an outer loop to find the hyperparameters of the model, which resulted in over 5 hours of training for a relatively small dataset, despite being trained with only 277 handmade sizing solutions. In [42] another example of predicting sizes of an amplifier given its target specifications is proposed, but the test to validate the results obtained was performed with only 10 samples from the original dataset, which is extremely low, and the performance and usability of the proposed model was not evaluated for new target specifications.

2.4.2 Reinforcement learning

Reinforcement learning (RL) aims to develop an agent that learns how to behave in a certain environment where the only feedback given is the reward of its actions. This interaction between agent and environment is depicted in Fig. 2.7. The primary goal of the agent is to maximize the notion of cumulative reward regarding its actions. These systems are capable of teaching robots to learn motor skills [44] or master complex board games like chess or Go [45].

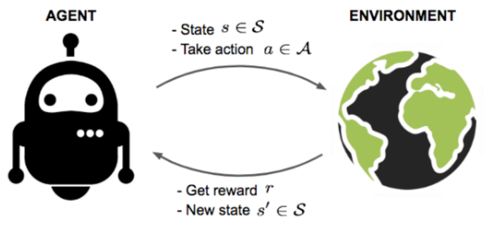

AGENT
ENVIRONMENT
- State $s \in \mathcal{S}$
- Take action $a \in \mathcal{A}$
- Get reward r
- New state $s' \in \mathcal{S}$

Figure 2.7 - Interaction between agent and environment. Picture taken from [46].

RL can also be used in analog/RF sizing. In [47] an agent learns from trial and error how to behave like a circuit designer evolving itself to finally discover circuit sizes that satisfy the performance specifications. Another instance where RL is used for sizing is in [48], a tool named AutoCkt, a ML optimization framework trained using deep RL, that is capable of finding post-layout circuit parameters for a certain parameter specification and can also acquire knowledge about the entire circuit design space using a sparse subsampling technique.

2.5 Conclusion

In this chapter, different methods used in circuit optimization sizing were introduced and compared regarding the evaluation engine used, with emphasis on simulation-based sizing. Using a simulator as the evaluation engine is the most widely accepted approach, with its main advantages being its generality and easy-and-accurate model. The main problem with this method is how time-consuming it is. In recent works, the use of ML tries to address and solve this problem by introducing ANNs into the optimization phase by either replacing or complementing the circuit simulator, as shown in Table 2.3.

In this work, the popular and powerful usage of an ANN to enhance the optimization-based sizing by complementing the circuit simulator will be applied. This method will be used in circuit topologies more complex than the ones previously discussed in the literature, and, despite the marginal loss of accuracy when compared with the circuit simulator, the speed-up gained using this approach will surely boost the optimization performance.

Table 2.3 - Speeding-up Simulation-based Sizing with ANNs Overview

Reference	Speed-up factor (up to)	Maximum Error	Number of Layers	Method
G. Wolfe, 2003 [33]	≈2000	60%	3	Complement/Replace simulator
T. O. Çakıcı, 2020 [34]	29.7	1.55%	3	Replace simulator
G. İslamoğlu, 2019 [35]	2.8	0.77%	4 & 5	Semi-replace simulator
K. Hakhamaneshi, 2019 [36]	n/r	n/r	4	Replace simulator
G. Alpaydin, 2003 [37]	n/r	n/r	3	Complement simulator
Hongzhou Liu, 2002 [38]	n/r	≈10000%	3	Replace simulator

n/r – not reported

3 PVT Corner Performance Estimator

Considering the main goals of this work and the related work presented and discussed in Chapter 2, it is proposed the elaboration of a PVT corner performance estimator using deep ANNs, complementing the circuit simulator to be used during sizing optimization. The ANNs will receive as input the performance figures respective to the typical conditions, i.e., TT conditions, obtained via accurate circuit simulation and the candidate circuit sizing solutions, and will predict the performances for the remaining PVT corners. In the optimization-loop of the simulation-based sizing, the PVT estimator will be located after the circuit simulator, as depicted in Fig. 3.1.

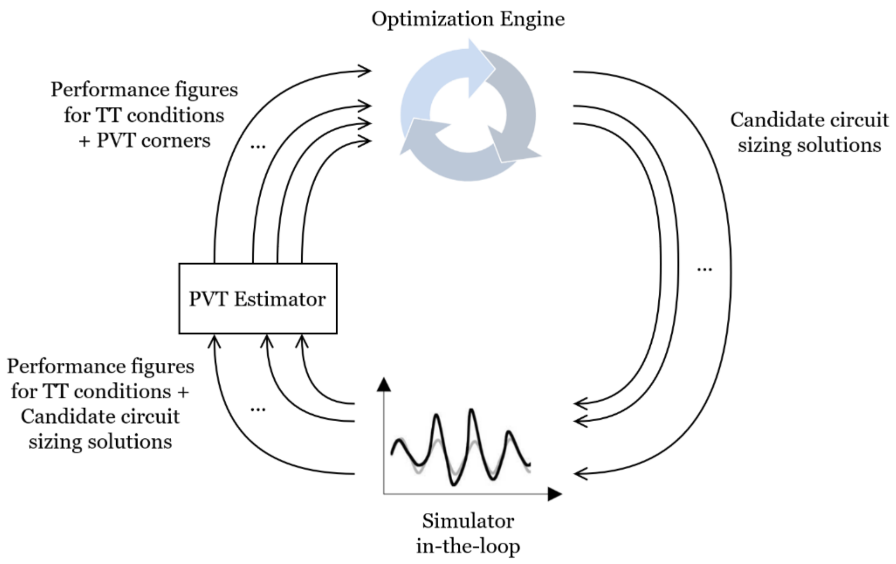

Optimization Engine
Performance figures for TT conditions + PVT corners
...
Candidate circuit sizing solutions
...
PVT Estimator
Performance figures for TT conditions + Candidate circuit sizing solutions
...
Simulator in-the-loop

Figure 3.1 - Location of the PVT estimator in the AIDA optimization loop

In the following sections, a brief discussion will be carried regarding the first case study and dataset that will be used in this work, and afterwards the proposed structure of the ANN will be presented.

3.1 Case Study

The development of the proposed tool will take part in the sizing of a complex dual-mode class C/D VCO presented in [7], which is presented in Fig. 3.2. In that work, instead of achieving the desired performance parameters with sequential SOOs, a single many-objective sizing optimization, described as “everything at once” optimization, is proposed to achieve the best performance boundaries while finding the optimal tradeoffs. The circuit simulator performed a multi-corner analysis and the optimization followed a worst-case corner criteria on top of a worst-case tuning range optimization, taking into account two different tuning modes, $b_{0000}$and b_{1111}. The results pushed the circuit to its performance limits, reducing to almost half of the power consumption of the original design and showed its potential for ultralow-power with more than 93% reduction.

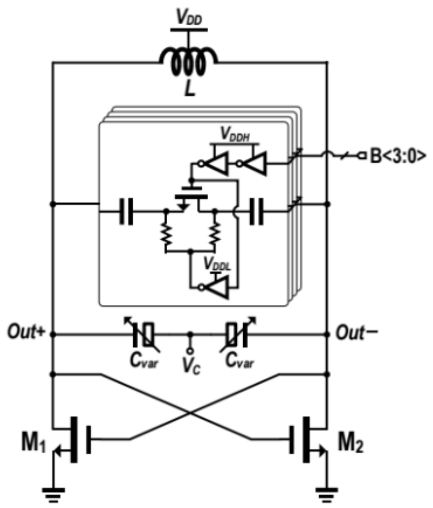


Figure 3.2 - Dual-mode class-C/D VCO schematic. Reprinted from [7].

In the optimizations carried, there were 28 optimization variables that affect the sizing of 43 devices. The full list can be found in Table 3.1.

Table 3.1 - Optimization variables

Variable	Units	Min.	Grid	Max.
ind_radius	µm	15	5	90
ind_nturns	-	1	1	6
ind_spacing	µm	2	1	4
ind_width	µm	3	1	30
mccl, m1l	nm	60	20	240
mccw, m1w	µm	0.6	0.2	6
mccnf, m1nf	-	1	1	32
mccm	-	1	1	100
moscapw	µm	0.4	0.2	3.2
moscapl	µm	0.2	0.2	3.2
mimvw, mimvl, mim1w	µm	2	0.2	20
r1l, r2l, r3l, r4l	µm	1	0.2	10
r1m, r2m, r3m, r4m	-	1	1	20
nfn1, nfn2, nfp1, nfp2	-	1	1	100

A total of 18 performances were considered and three optimizations were performed with populations of 512 elements optimized for 1000 generations. Of all the sizing solutions, the POFs of the three optimizations provided, in total, 769 optimal sizing solutions. Each optimization took approximately 100 hours to complete in an Intel-Xeon CPU E5-2630-v3@2.40 GHz with 64 GB of RAM workstation using eight cores for parallel evaluation, resulting in 300 hours total, i.e., more than 12 days. Once again, the main goal of the proposed PVT Estimator in this dissertation will be to reduce this execution time to an acceptable range.

3.2 Dataset

The source of the dataset will be simulated performances and associated sizing parameters generated by an optimization-based sizing of the circuit in the previous section. In total, 9 different testbench variations will be considered (TT, FF, FS, SF, SS, 300mV, 400mV, m40dC and 85dC) that produce 10 different performance figures each, and, due to the worst-case tuning range optimization (two tuning modes are evaluated, $b_{0000}$and b_{1111}), each sizing must be simulated 18 times, providing a total of 180 simulated performance figures. The full list of testbench variations can be seen in Table 3.2 and the list of performances in Table 3.3.

Table 3.2 - List of testbenches variations: TT and PVT corners

Name	Process	Voltage	Temperature
TT	TT	0.35 V	25ºC
FF	FF	0.35 V	25ºC
FS	FS	0.35 V	25ºC
SF	SF	0.35 V	25ºC
SS	SS	0.35 V	25ºC
300mV	TT	0.3 V	25ºC
400mV	TT	0.4 V	25ºC
m40dC	TT	0.35 V	-40ºC
85dC	TT	0.35 V	85ºC

Table 3.3 - List of performances considered for TT and PVT corners

Measure	Units	Description
f_{osc}	GHz	Oscillation frequency
PN@10kHz	dBc/Hz	Phase noise at 10KHz
PN@100kHz	dBc/Hz	Phase noise at 100KHz
PN@1MHz	dBc/Hz	Phase noise at 1MHz
PN@10MHz	dBc/Hz	Phase noise at 10MHz
power	mW	Power consumption
FOM@10kHz	dBc/Hz	Figure-of-merit at 10KHz
FOM@100kHz	dBc/Hz	Figure-of-merit at 100KHz
FOM@1MHz	dBc/Hz	Figure-of-merit at 1MHz
FOM@10MHz	dBc/Hz	Figure-of-merit at 10MHz

The dataset will have a total number of 48 features, where 28 of them are the optimization variables plus 20 performance figures of the simulation in TT conditions in two different modes, i.e., $b_{0000}$and b_{1111}, and a total of 160 labels, i.e., the performance figures of the remaining corner variations in two different tuning modes.

All this raw data has to be pre-processed before using it in the training phase of the model, so some feature engineering will be needed. The dataset contains small input device sizes (in the order of nanometers) combined with other optimization variables, which can be simple integers (for example, ind_nturns), and with performance figures, which can have large values in the order of gigahertz (for example,

$oscillation\,frequency)$. This combination causes the learning algorithm of the ANN to wrongly compute the weights associated with these small values, almost completely negating their influence on the output. To solve this problem, data normalization will be performed on the entirety of the dataset. Two methods can be used to achieve this: standardization and normalization. Standardization scales the values while considering standard deviation, which is beneficial to reduce the effect of outliers in the data. Normalization scales all values to a fixed range between, for example, 0 and 1. This scaling does not alter the features' distribution, and because of the decreased standard deviations, the effect of the outliers increases. The formulas used for standardization and normalization are shown in (3.1) and (3.2), respectively, where μ represents the mean value and σ the standard deviation.

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

$$z = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.2}$$

The dataset will present some null performance values. These values won't be fed to the network because they will affect the performance of the model, so firstly they must be removed. This technique in feature engineering is called imputation. Checking and removing duplicated rows in the dataset must also be performed due to its negative effect on the performance of the model.

Another pre-processing approach to be taken will be handling outliers. This can be done using visual or statistical methodologies. Detecting outliers visually implies checking in a graph for outliers for each feature of the dataset and due to the size of this dataset, i.e., the high number of labels, this task will likely be performed using statistical methodologies. Outlier detection using standard deviation will be conducted, which simply classifies a value as an outlier if its distance to the average value of the feature is higher than $x\ \times standard\ devation$. This can also be done using percentiles.

Although many other techniques of feature engineering can be performed on the dataset, the previously mentioned ones will be the most important to use. After this, the dataset is ready to be fed to the network.

3.3 ANN Development

In this section, an overview of the ANN structure and training phase will be presented followed by its tuning process.

3.3.1 Structure (In/Out)

ANNs are structured in *n* + 2 layers: a single input layer, a single output layer and *n* hidden layers as shown in Fig. 3.3. Each layer has a certain number of neurons.

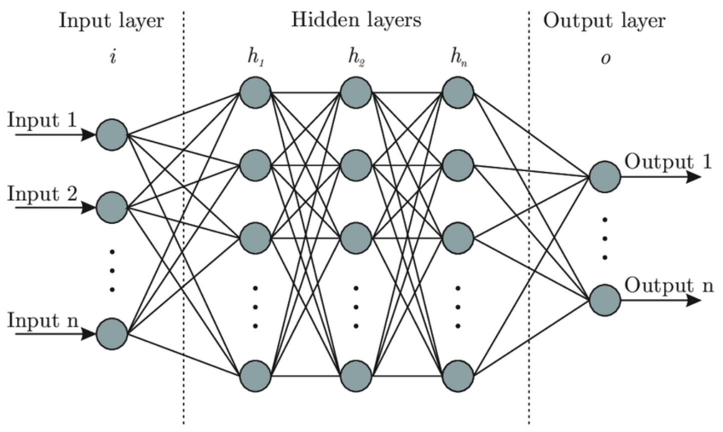


Figure 3.3 - Example of an ANN's structure. Reprinted from [49].

The input layer has the same number of neurons as the number of features selected for each data point in the dataset. Likewise, the output layer has the same number of neurons as the number of outputs desired. Each neuron is essentially a function that outputs y as a function of its inputs $x_1, \ldots, x_n$ as depicted in Fig. 3.4 and demonstrated by:

$$y = \phi\left(\sum_{i=1}^{n}(w_i x_i) + b\right), \tag{3.3}$$

where ϕ is the neuron's activation function, w_i is the weight associated with the input x_i and b the neuron's bias which is basically treated as the weight, w_0, of the constant input value $x_0 = 1$. (3.3) can be rewritten as a simpler version:

$$y = \phi\left(\sum_{i=0}^{n}(w_i x_i)\right) \tag{3.4}$$

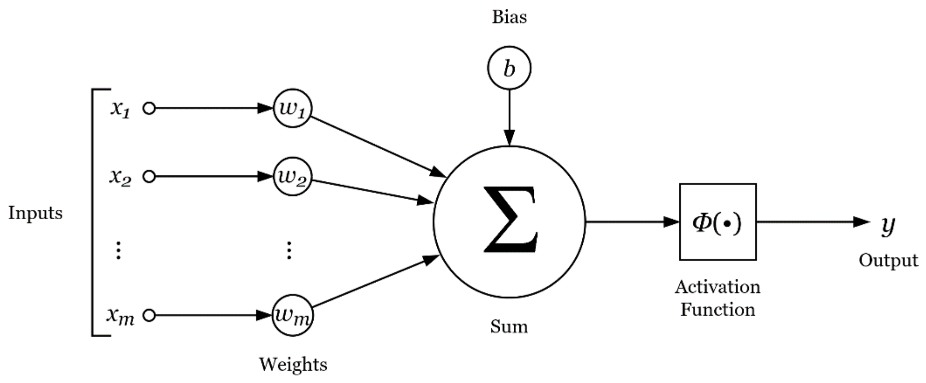


Figure 3.4 - Artificial Neuron

For the case of fully connected ANNs, each neuron in a hidden layer j will have as inputs each output of the previous layer $j - 1$, and its own output will be an input of each neuron in layer $j + 1$. If preceded by a layer with k neurons, the output of a certain layer j with m neurons can be described as a $\mathbb{R}^k \rightarrow \mathbb{R}^m$ function where the output of each neuron is described by (3.4).

Using the architecture of the fully connected ANN in Fig. 3.3 as an example, each neuron receives a certain number of inputs, multiplies each one of them by its corresponding weight, sums them all, passes the result by an activation function and outputs it. The final output is a combination of the input's propagation through each layer of the network. The weights of each neuron are what the ANN needs to learn.

In this work, the structure of the ANNs used will be similar to the one depicted in Fig. 3.3, i.e., it will be fully connected. Each ANN will estimate the performance figures of a specific corner for a specific tuning mode, so the output layer will have 10 neurons. Each ANN will receive as inputs, the sizing of the circuit, which consists of 28 optimization variables, and 10 performance figures of the simulation in TT conditions for its corresponding tuning mode, which means that the input layer will have a total 38 neurons. The number of hidden layers and number of nodes per hidden layer will be determined in the tuning phase. A trial and error approach will be taken to find the best possible solution to these two hyperparameters (further explained in Section 3.3.3). The structure of the ANN implemented for corner FF and tuning mode b_{0000} is shown in Fig. 3.5.

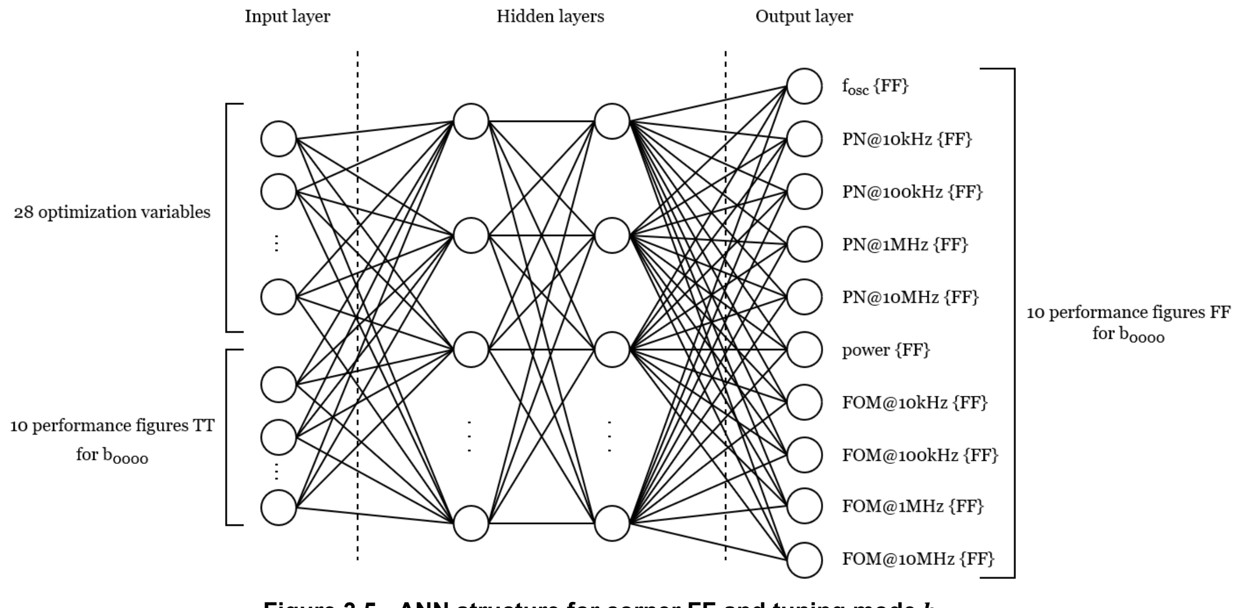


Figure 3.5 - ANN structure for corner FF and tuning mode b_{0000}

Usually, the only linear activation function used in a regression-based ANN is on the output layer and the most used is the linear function $f(z) = z$, depicted in Fig. 3.6. The output of the regression model in this work is going to be a real value, positive or negative, so the simplest function to reach the entire real set is a linear function.

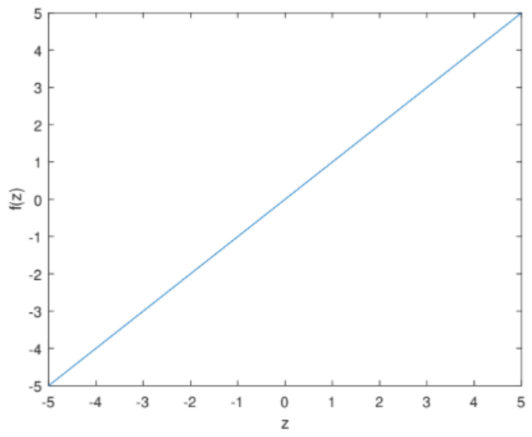


Figure 3.6 - Linear function $\mathbf{f}(\mathbf{z}) = \mathbf{z}$

3.3.2 Training Phase

Loss Function and Backpropagation

The learning phase of the ANN is an iterative process where the weights of each neuron are updated using the backpropagation algorithm.

In each iteration, a certain input x is fed to the network and the obtained output $\hat{y}$ is compared to the desired output y computing the error (or loss) J between them. Because this is a regression problem, the most appropriate loss function is the Mean Squared Error (MSE). This loss is calculated by (3.5).

$$J = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.5}$$

The derivate of this error is propagated backwards through the network and each weight is updated, according to (3.6), using this value. In (3.6) and (3.7), w_i represents a certain weight, w_{i+1} represents the weight after the update and η represents the learning step.

$$w_{i+1} = w_i - \Delta w \tag{3.6}$$

$$\Delta w = \left(\eta \frac{\partial J}{\partial w} w_i\right) \tag{3.7}$$

The training process of the ANN corresponds to an optimization problem solved using a method called gradient descent that iteratively finds the weights that minimize the loss function of the model.

Optimizer

To speed-up the training phase of the ANN one can use faster optimizers than the gradient descent. These variants of the gradient descent algorithm try to reduce the total number of iterations needed to reach the minimum solution of the loss function making the convergence less time-consuming. The most popular optimizer nowadays is the Adaptive Moment Estimation. This method is exceptionally appropriate for sparse input data due its adaptive learning rate, while having a fast convergence. Because of these reasons, this will be the chosen optimizer for the model in this work. If this method doesn't prove to be helpful while testing the performance of the model, Root Mean Square Prop will be used instead.

3.3.3 Model Fine Tuning

One problem of ANNs is the vast number of hyperparameters to be tuned, which to find the best values one must search them by trial and error. In this section, various hyperparameters are presented, indicating their influences on the network and how their best values will be found. Finally, a common problem encountered when training ANNs called Overfitting is presented along with its solutions.

Number of Hidden Layers and Neurons

The number of hidden layers directly influences the complexity of the ANN model. A large number of layers results in a more complex function that maps an input to its output and tends to lead to better results. However, it also increases the amount of computational power required and time spent to train the ANN. By contrast, a small number of hidden layers might lead to an ANN that is not capable of modelling a problem accurately. Even though more layers lead to better results, this can lead to overfitting the model to the training data. This combined with the reasons mentioned before, makes it so one cannot use too many layers in an ANN. There is no formula nor perfect method to choose the number of layers in a network. The used approach in this work will consist in testing and comparing, by trial and error, the results of the network while gradually increasing the number of layers until the accuracy of the model cannot further improve.

While choosing the best number of layers for the network, one also must consider the number of neurons on each one of them. For the input and output layers this choice is simple, for each feature considered there must be one neuron in the input layer, and for each expected output there must be one neuron in the output layer. For the hidden layers, one must, once again, find the best number of neurons using a trial and error process. Many neurons in a hidden layer leads to more complex models which ultimately increases greatly the training time of the network due to the increase in amount of weights computed.

Activation Functions of the Hidden Layers

For the hidden layers of the ANN, one should use non-linear activation functions to increase its capability to model highly complex relationships between its input and output. The following functions will be experimented in the model, and their results will be compared to choose the most adequate for this work.

The sigmoid function, shown in Fig. 3.7, can map the input values of a neuron to an output range of $]0,1[$. This prevents cases where the output of the neuron reaches very high values which can happen when using an unbounded linear function. Another quality of this function is its sensibility to input changes in the region near $z = 0$ which results in a good separation of data. However, the responsiveness of the function starts to decrease when dealing with bigger input values, reaching almost a gradient value of 0, therefore the model can no longer update properly.

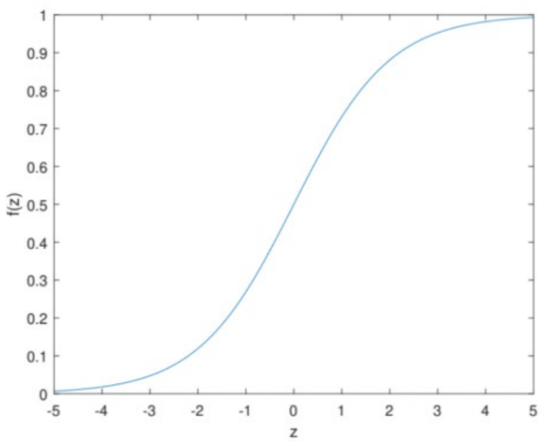


Figure 3.7 - Sigmoid function

One of the most popular activation functions nowadays is the Rectified Linear Unit (ReLU) function [50], shown in Fig. 3.8, computed through $f(z) = \max(0, z)$. The output value of the ReLU function is equal to the input value for inputs greater than 0 and equal to 0 for all the other values. This non-linear function requires less computational power than the sigmoid function due to its simpler formula. However, the function is not bounded for positive input values making it susceptible to exploding output values. Despite this, it solves the low gradient value encountered in the sigmoid function. The main problem appears in the negative input value region, where the gradient equals to 0 which will stop any network update from happening (regarding the corresponding neuron). This problem is called the dying ReLU problem.

To solve this problem, some variations to the ReLU were developed, such as the leaky ReLU [51] or the Exponential Linear Unit (ELU) [52]. The ELU introduces a smooth derivate for the negative input values and is determined by (3.8), where α is a parameter to be tuned. This function is depicted in Fig. 3.8.

$$f(z) = \begin{cases} \alpha(e^z - 1) & if\ z \leq 0 \\ z & if\ z > 0 \end{cases} \tag{3.8}$$

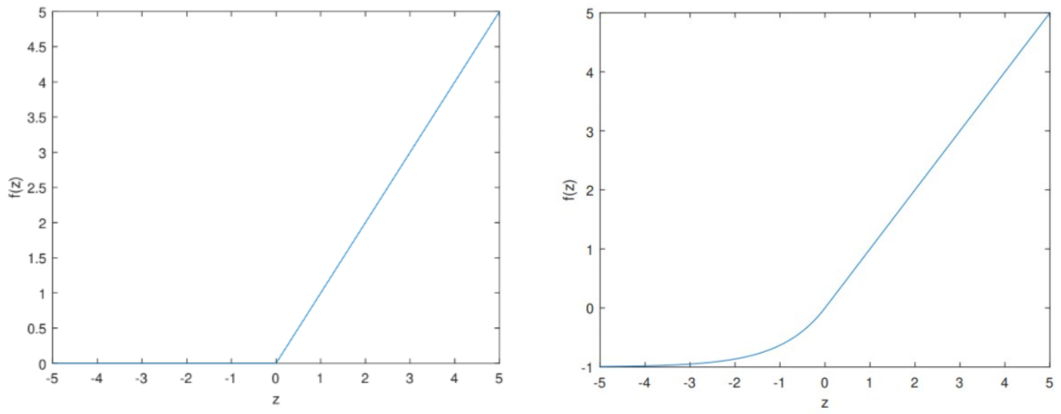


Figure 3.8 - ReLU function (left) and ELU function (right)

Learning Rate

This hyperparameter dictates the impact of the error gradient each time a weight of the ANN is updated. If a high value is chosen for this parameter, the optimization algorithm may find a sub-optimal set of weights, or the training process becomes unstable which, worst-case scenario, can make the algorithm diverge. A low value of this parameter can make the training process too slow where, if given a small number of epochs, the algorithm doesn't reach a minimum. Both cases are depicted in Fig. 3.9.

The learning parameter is one of the most important hyperparameters when training an ANN due to its big influence on the convergence of the optimization algorithm. For this work, the tuning process of this hyperparameter will be a trial and error approach, starting with a big value and iteratively reduce it until no improvement is reached. The chosen value will be the one with the highest value with the best results due to having the lowest training time.

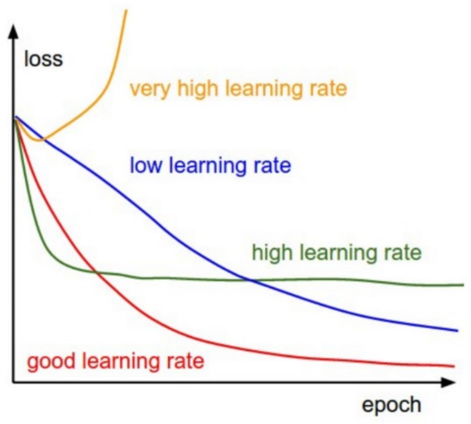


Figure 3.9 - Evolution of loss function with different learning rates. Reprinted from [53].

Overfitting and Regularization

Overfitting is one of the main problems that one can encounter when training a ANN. It refers to a model that models the training data too well, i.e., the model learns the detail and noise in the training data to the point that it negatively impacts the performance of the model on new unseen data. Random fluctuations and noise present in the training data do not apply to new data so it negatively affects the ability of the model to generalize.

To solve this problem, a regularization technique can be used when training the ANN. The techniques that are going to be tested in this work are Early Stopping [54] and Dropout [55].

With Early Stopping, the model will remain in the training phase until there is no improvement in the validation set in two consequent epochs, causing the training to stop, avoiding overfitting.

Dropout is becoming one of the most popular regularization techniques because it solves two important problems in an ANN: it prevents overfitting of the model and provides a method of approximately combining different ANN architectures. Dropout consists in removing temporarily certain neurons from the ANN along with its incoming and outgoing connections.

3.4 Conclusions

In this Chapter, the proposed solution for the optimization sizing speed-up was presented. First, the main case study of this work was described, which consists in an optimization sizing of a class C/D VCO, and its corresponding dataset, that will be used to train the ANNs of the PVT Estimator, was defined, where the optimization variables and performance parameters were shown following a small outline of the pre-processing necessary for the dataset. Finally, the ANNs structure was presented and the tuning phase of its hyperparameters was explained in detail.

4 Results Pre-Integration

Considering the strategy that will be used to build the ANN, defined in Chapter 3, and all the different parameters to be tuned, the following subsections present the work regarding the programming of the ANN and its expected results are discussed. In order to do this, the dataset of the complex dual-mode class C/D VCO circuit optimization, previously defined in Chapter 3, will be used.

Throughout this part of the work the language used to program the ANN was Python, using both Tensorflow [56] and Keras [57] as ML libraries.

The starting point of the ANN architecture is the one described in Section 3.3.1, where the output layer contains 10 nodes, one for each performance parameter of a certain combination of tuning mode and PVT corner variation. The optimization was performed for 9 testbench variations which 8 represent the PVT corners, and 2 tuning modes, and thus, in total, 16 different ANNs will be required.

4.1 Tuning Phase

In this section the necessary pre-processing applied to the dataset will be described followed by the tuning parameters phase.

Dataset

The dataset contains 92115 data entries composed by, as described in Section 3.2, 48 features where 28 represent the optimization variables and the other 20 represent the TT performance figures, 10 for each tuning mode. As for labels, the dataset contains 160 performance figures of the remaining PVT corner variations in two different tuning modes.

For each different ANN, it is only needed the performance figures of one combination of corner variations and tuning mode, so firstly the dataset had to be divided in 16 different datasets where each dataset represents a different corner combination. To increase model accuracy, only the TT performance figures that represent the same tuning mode as the labels are kept, so the final dataset structure only contains 38 features and 10 labels.

Some data entries have *null* values on the features and/or labels, representing sizing solutions that the simulator couldn't produce a meaningful performance figure. These entries had to be removed from each dataset to provide the best possible data to the ANNs along with duplicated rows. The division of the original dataset into smaller datasets and the removal process explained before is depicted in Table 4.1 and Table 4.2, where some entries of the original dataset and dataset for FF b_{0000} are shown, respectively.

Table 4.1 - Structure of the original dataset

Entry #	Features (48)			Labels (160)			
	Design variables	Performances TT b_{0000}	Performances TT b_{1111}	Performances FF b_{0000}	…	Performances m40dC b_{1111}	Performances 85dC b_{1111}
0	values ✓	values ✓	values ✓	null	…	values ✓	null
1	values ✓	values ✓	values ✓	values ✓	…	null	null
2	values ✓	null	null	values ✓	…	null	null
3	values ✓	values ✓	values ✓	null	…	null	null
…	…	…	…	…	…	…	…
92114	values ✓	values ✓	values ✓	values ✓	…	values ✓	values ✓

Table 4.2 - Structure of the dataset for FF b_{0000}

Entry #	Original entry #	Features (38)		Labels (10)
		Design variables	Performances TT b_{0000}	Performances FF b_{0000}
0	1	values ✓	values ✓	values ✓
1	3	values ✓	values ✓	values ✓
…	…	…	…	…
83037	92114	values ✓	values ✓	values ✓

Finally, the outliers present in each dataset must be removed. To do this, for each performance figure, the 1% lowest and highest values were cut from the dataset, alongside their entire row of data. The final sizes of the dataset for each PTV corner ANN can be found in Table 4.3.

Table 4.3 - Dimensions of datasets for the training of the different ANNs

Corner	Tuning mode			
	b_{0000}	% total	b_{1111}	% total
FF	81377	88.34	79300	86.09
FS	81842	88.85	81200	88.15
SF	82247	89.29	76794	83.37
SS	73456	79.74	48742	52.91
300mV	77608	84.25	69017	74.92
400mV	81865	88.87	81342	88.30
m40dC	81182	88.13	77103	83.70
85dC	80707	87.62	82028	89.05

Finally, all the 16 datasets were randomized, and split into two datasets:

- Training dataset: 90% of the original dataset;
- Test dataset: Remaining 10% of the original dataset.

The training dataset will be used to train the ANNs while the test dataset will be used to test the models. As for the tuning of the ANN architecture phase, the tuning parameters phase was only performed in the ANN regarding the corner FF with tuning mode b_{0000}.

Tuning parameters

With the dataset ready, the ANN's tuning parameter phase starts, as addressed in Section 3.3.3. As evaluation methods for the ANN's training, three following metrics were used. MSE, defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.1}$$

Mean Absolute Error (MAE), defined by:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - x_i| \tag{4.2}$$

And finally, Mean Absolute Percentage Error (MAPE), defined by:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{4.3}$$

Due to the nature of the values that are trying to be predicted with this ANN, the error between predicted value and actual value must be small. Because of this, an error value lower than 1% (for the case of the MAPE) was chosen as a reasonable target to achieve for the accuracy of the ANN.

Despite having three different metrics as evaluation methods, the loss function of the ANN throughout the tuning phase was the MSE. As stated in Section 3.3.2, this is one of the most popular loss functions in regression problems like the one addressed in this dissertation. A similar approach was used to choose the optimizer, with Adam being pointed as the most popular method.

Regarding the batch size of the training phase, 128 was chosen as an appropriate value considering the size of the training dataset used.

To solve the problem regarding the different magnitudes in the dataset explained in Section 3.2, normalization for a range of 1 to 2 was used. This range was chosen, over the typical 0 to 1, due to its influence in MAPE values, making them explode in value due to values close to 0 in the denominator of the MAPE formula in (4.3).

The tuning phase of the ANN follows the strategy stated in Section 3.3.3 and the first parameters to be determined were: the number of layers, number of neurons per layer, and learning rate.

To determine the adequate number of layers for the model, two studies were made, the first one using 2 hidden layers for the model, and the second 3 hidden layers. Table 4.4 presents all the different parameters and their values considered in the 2 hidden layers study.

Table 4.4 - Different parameters and corresponding values to study

Size of hidden layer 1	Size of hidden layer 2	Learning rate
200, 320, 440, 560	200, 300, 400, 500	0.00005, 0.0001, 0.0005, 0.001

It is common practice in ANN development that the size of a hidden layer is always equal or larger than the following hidden layer. Considering this, the different combinations of these parameters were studied, and their results are shown in table 4.5.

The lowest values of each metric (MSE, MAE and MAPE) at both training and test phases were highlighted, and achieving 5 of the 6 total lowest metrics, the best combination of these parameters is:

- Hidden layer size 1: 440 neurons
- Hidden layer size 2: 400 neurons
- Learning rate: 0.0001

Given these results, a comparison with 3-hidden layer ANN architecture is carried. In this next study, the learning rate was set to 0.0001 considering it was the best value of the previous study. In table 4.6 is shown the different parameters and the values that were considered. This study follows the same rule as the previous one regarding the sizes of the hidden layers and its results are shown in table 4.7.

When comparing the results of the two studies, it is clear that there is no improvement when increasing the number of hidden layers. The best error results with 3 hidden layers are 18% to 30% higher than the best results with 2 hidden layers. Considering this fact there is no need to increase the number of hidden layers of the ANN, so no further study is required. These values will be used for the remaining of the tuning phase.

Table 4.5 - Results of hidden layers size/learning rate study.
Using activation function of hidden layers: ReLU, dropout rate: 20%

Hidden layers		Learning rate	Training loss			Test loss		
#1	#2		MSE(x10^{-4})	MAE(x10^{-3})	MAPE	MSE(x10^{-4})	MAE(x10^{-3})	MAPE
200	200	0.001	2.0271	9.0719	0.5865	2.9237	9.6342	0.6262
		0.0005	1.6593	8.4255	0.5503	2.6371	8.9969	0.5903
		0.0001	0.9633	4.9812	0.3263	2.0497	5.6614	0.3736
		0.00005	1.5778	6.3185	0.4126	2.4236	6.9236	0.4543
320	200	0.001	1.6222	8.4173	0.5433	2.5826	9.0382	0.5867
		0.0005	1.2829	7.7522	0.5101	2.1944	8.4498	0.5579
		0.0001	0.7502	4.2473	0.2801	1.7580	4.9959	0.3316
		0.00005	1.1752	5.3409	0.3510	1.9515	5.9199	0.3911
320	300	0.001	1.5518	7.4064	0.4838	2.4660	7.9911	0.5243
		0.0005	0.9664	6.0647	0.3905	1.8117	6.7411	0.4368
		0.0001	0.6854	4.0345	0.2649	1.6603	4.8014	0.3176
		0.00005	1.5619	6.6477	0.4345	2.5805	7.3492	0.4827
440	200	0.001	0.9685	5.9367	0.3893	1.7024	6.5298	0.4304
		0.0005	0.9217	5.7347	0.3669	1.7494	6.4228	0.4139
		0.0001	0.6052	4.0061	0.2625	1.4736	4.7756	0.3149
		0.00005	1.2531	6.3213	0.4136	2.2113	7.0737	0.4648
440	300	0.001	2.0921	8.0931	0.5290	3.4045	8.7485	0.5760
		0.0005	0.9115	6.3530	0.4114	1.7763	7.0825	0.4612
		0.0001	0.6586	4.0705	0.2677	1.4629	4.7683	0.3155
		0.00005	1.2429	6.0380	0.3935	2.1168	6.7248	0.4407
440	400	0.001	1.1995	6.0374	0.4000	2.1117	6.6899	0.4457
		0.0005	0.6916	4.7748	0.3087	1.5629	5.4907	0.3578
		0.0001	**0.5372**	**3.5950**	**0.2344**	1.5345	**4.4429**	**0.2919**
		0.00005	1.1603	5.4496	0.3574	2.3146	6.2074	0.4093
560	200	0.001	1.0060	6.2592	0.4213	1.9085	6.9229	0.4667
		0.0005	0.7292	5.0753	0.3370	1.5336	5.8173	0.3874
		0.0001	0.5895	3.8783	0.2558	1.5572	4.6842	0.3110
		0.00005	0.9646	5.0136	0.3267	1.9388	5.7902	0.3798
560	300	0.001	0.9744	5.2125	0.3425	2.1092	5.9488	0.3937
		0.0005	0.7596	5.2312	0.3366	1.7456	5.9887	0.3887
		0.0001	0.5920	3.9961	0.2601	1.4898	4.8121	0.3157
		0.00005	0.8541	4.6235	0.3033	1.6815	5.3325	0.3516
560	400	0.001	5.8195	14.5366	0.9489	6.4470	14.9806	0.9800
		0.0005	0.8302	5.4530	0.3566	1.6963	6.1971	0.4072
		0.0001	0.5769	3.8253	0.2504	1.6146	4.6759	0.3085
		0.00005	0.8927	4.9385	0.3201	1.7218	5.6835	0.3707
560	500	0.001	1.5522	7.6797	0.5064	2.4262	8.3103	0.5505
		0.0005	0.5916	4.2945	0.2834	**1.4161**	5.0691	0.3365
		0.0001	0.6000	4.5236	0.3017	1.5613	5.3601	0.3586
		0.00005	0.8585	5.3343	0.3411	1.7989	6.1322	0.3955

Table 4.6 - Different parameters and corresponding values to study

Size of hidden layer 1	Size of hidden layer 2	Size of hidden layer 3
200, 320, 440	200,300,400	200,300,400

Table 4.7 - Results of hidden layers size study
Using activation function of hidden layers: ReLU, learning rate: 0.0001 and dropout rate: 20%

Hidden layers			Training loss			Test loss		
#1	#2	#3	MSE(x10^{-4})	MAE(x10^{-3})	MAPE	MSE(x10^{-4})	MAE(x10^{-3})	MAPE
200	**200**	**200**	1.9722	8.7683	0.5709	3.1234	9.4508	0.6186
320	**200**	**200**	0.9191	5.8708	0.3796	2.0354	6.7268	0.4385
320	**300**	**200**	1.5984	7.7507	0.5039	2.8096	8.5406	0.5586
320	**300**	**300**	0.9735	6.1391	0.3957	2.0810	7.0175	0.4560
440	**200**	**200**	**0.7186**	**4.4182**	**0.2897**	**1.7230**	**5.2482**	**0.3465**
440	**300**	**200**	0.9799	5.9074	0.3852	1.9675	6.6465	0.4361
440	**300**	**300**	0.8321	5.6166	0.3613	1.8117	6.4913	0.4210
440	**400**	**200**	1.0870	6.4679	0.4147	2.1437	7.2971	0.4718
440	**400**	**300**	1.1155	6.7236	0.4337	2.4101	7.6854	0.4997
440	**400**	**400**	0.8439	5.5525	0.3570	1.8376	6.4363	0.4172

The next parameter to be tuned is the activation function of the hidden layers. As stated in Section 3.3.3, the different activation functions considered in this study were: sigmoid, ReLU, leaky ReLU and ELU. The results are shown in Table 4.8.

Table 4.8 - Results of activation function study
Using 2 hidden layers of sizes 440 and 400, learning rate: 0.0001 and dropout rate: 20%

Activation function	Training loss			Test loss		
	MSE(x10^{-4})	MAE(x10^{-3})	MAPE	MSE(x10^{-4})	MAE(x10^{-3})	MAPE
Sigmoid	3.8048	10.4705	0.6847	4.2671	10.8543	0.7120
ReLU	**0.5372**	**3.5950**	**0.2344**	1.5345	**4.4429**	**0.2919**
leaky ReLU	0.8000	4.2949	0.2804	**1.4946**	4.8755	0.3204
ELU	2.8294	8.4947	0.5537	3.4148	8.9600	0.5855

As can be seen by the highlighted values, the activation function that generates the lowest error in almost all cases is the ReLU function, proving why it is one of the most popular activation functions when working with ANNs nowadays [50]. Leaky ReLU has the best MSE test loss of all 4 activation functions, but in the remaining values falls short to the ReLU. Regarding the sigmoid and ELU functions, these demonstrated weak results when comparing with the other 2 functions, getting in all metrics, values 2 times worse. This study proved that the best activation function for this work is the ReLU function, which will be the function used in the final model.

The last parameter to be tuned is the dropout rate, which until now was set to 20%. The different values and corresponding results are shown in Table 4.9.

Table 4.9 - Results of dropout rate study
Using 2 hidden layers of sizes 440 and 400, learning rate: 0.0001 and activation function of hidden layers: ReLU

Dropout rate	Training loss			Test loss		
	MSE(x10^{-4})	MAE(x10^{-3})	MAPE	MSE(x10^{-4})	MAE(x10^{-3})	MAPE
0%	1.6501	8.4134	0.5456	2.3367	8.8968	0.5786
5%	0.5724	3.6952	0.2417	1.6456	4.6661	0.3074
10%	**0.5346**	3.8489	0.2511	1.6371	4.7203	0.3105
20%	0.5372	**3.5950**	**0.2344**	**1.5345**	**4.4429**	**0.2919**
30%	0.7887	4.8243	0.3167	1.7034	5.5186	0.3647

Analyzing the results, dropout rates of 5%, 10% and 20% show the best and relatively similar values between them while the values for dropout rate of 0% and 30% show a decrease in accuracy of the ANN. The worst results come from having no dropout rate at all which reveals the necessity of this regularization technique. A dropout rate of 20% was chosen for the final model, given that the ANN presents the best results with this value.

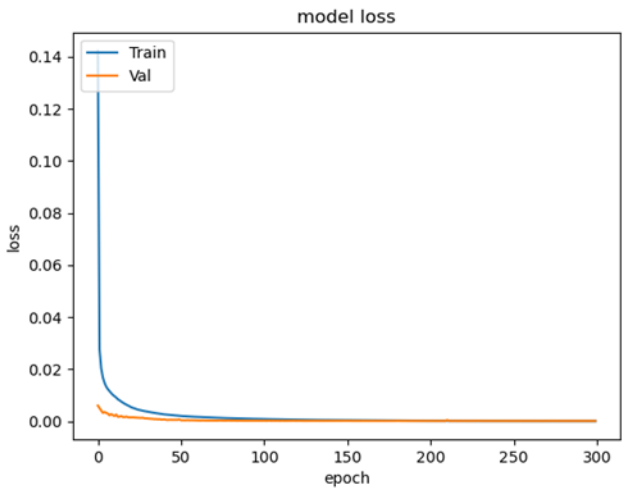
4.2 Final Model

All the parameters of the ANN are now tuned to achieve the best performance possible, and a brief summary of the model is shown in Table 4.10 along with the metrics of the training phase at each epoch.

Table 4.10 - Summary of ANN

Parameter	Value
Input layer size	38
Hidden layer 1 size	440
Hidden layer 2 size	400
Output layer size	10
Loss function	Mean Squared Error
Optimizer	Adam
Batch size	128
Hidden layers activation function	ReLU
Learning rate	0.0001
Dropout rate	20%
Training epochs	300
Validation split	20%

Analyzing Fig. 4.1 and Fig. 4.2, it is possible to state that the number of training epochs used for training is adequate due to the almost stagnation of both training and validation curves in the MAE and MAPE loss functions, and complete stagnation for the MSE loss function.



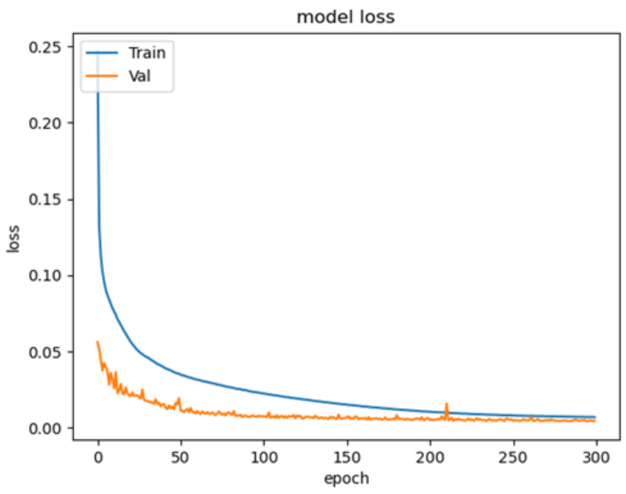


Figure 4.1 - MSE loss function (left) and MAE loss function (right)

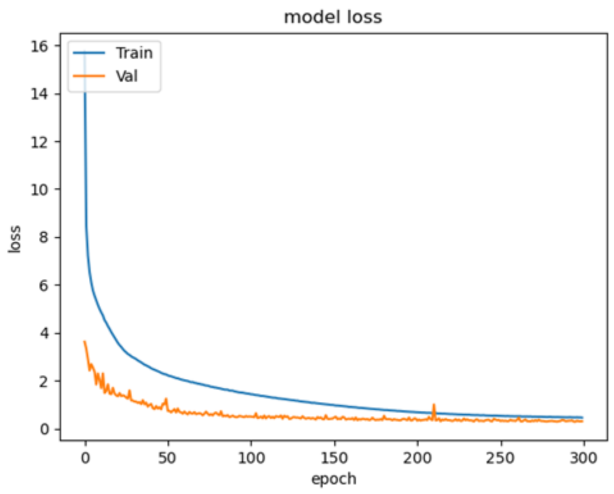


Figure 4.2 - MAPE loss function

The final values of the 3 metrics for all 16 ANNs are shown in Table 4.11. As can be seen by the MAPE values of both training and test loss, the final model in all ANNs is presenting better performance than the initial goal of 1% error.

Table 4.11 - Final model metric values

Tuning mode	Corner	Training loss			Test loss		
		MSE(x10^{-4})	MAE(x10^{-3})	MAPE	MSE(x10^{-4})	MAE(x10^{-3})	MAPE
b_{0000}	FF	0.5346	3.5950	0.2344	1.5345	4.4429	0.2919
	FS	0.5046	3.7093	0.2438	1.5623	4.3444	0.2873
	SF	0.7096	3.7298	0.2441	1.2777	4.2840	0.2810
	SS	0.9830	4.3566	0.2833	1.6931	4.9340	0.3236
	300mV	0.7763	5.3615	0.3524	1.5838	5.9087	0.3919
	400mV	0.6531	4.5662	0.3061	1.0625	4.9738	0.3339
	m40dC	0.6195	4.0734	0.2665	1.1569	4.3965	0.2890
	85dC	0.6881	4.4254	0.2965	1.1435	4.8061	0.3237
b_{1111}	FF	1.2872	5.4180	0.3638	2.3003	5.9296	0.3987
	FS	0.7244	4.0943	0.2727	1.7098	4.7493	0.3163
	SF	0.8058	4.3929	0.2858	1.7478	5.0881	0.3324
	SS	1.1053	7.8992	0.5777	2.8484	11.5884	0.8384
	300mV	1.8979	7.0264	0.4585	3.4236	8.1042	0.5304
	400mV	1.7348	7.4566	0.4828	3.1930	8.4154	0.5486
	m40dC	5.9536	9.3559	0.6163	8.4010	10.8045	0.7134
	85dC	1.0614	4.5612	0.3037	1.7748	5.1120	0.3418

4.3 Test Results

In order to check the performance of the ANN, the prediction errors of 4 different performances were obtained for the points that belong to the test. These performances are:

- Oscillation frequency;
- Power;
- Phase noise in 10MHz;
- Figure of Merit in 10MHz.

All 4 of these performances were obtained for corner FF in mode b_{0000}. In Tables 4.12, 4.13, 4.14 and 4.15 the results of the predictions are presented, where 5 different sets of results are shown:

- Best: 5 lowest MAPE values;
- 25th: 25th quantile value and the next 4 points (in order of MAPE);
- Median: median value and the next 4 points (in order of MAPE);
- 75th: 75th quantile value and the next 4 points (in order of MAPE);
- Worst: 5 highest MAPE values.

As observable on the results, the ANN is capable of achieving extremely accurate results. In all four performances the best results are almost perfect predictions of the true value with errors of 0.0005% at most. Another great indicator of the performance of the ANN is the quantile values. The worst 25th quantile value of the 4 performances is an error of 0.15% which indicates that 25% of all the predictions have a MAPE lower or equal for that performance (phase noise for this case). The medians of all 4 performances also show evidence of an accurate model ANN. The worst median value, also belonging to the phase noise performance, shows a small error of 0.28% between prediction and real value. The same can be said for the 75th quantile values, where the worst one sits at 0.42% for the phase noise again. Once again, these results show the powerful prediction capabilities of the model as it states that 75% of the points predicted have 0.42% or lower error values. Now finally the worst MAPE values are the extreme situations that the model encountered, specifically, mostly points that belong to a range of values very different than the most represent ones on the dataset. This can be seen for example in the worst MAPE values of the oscillation frequency where real values correspond to 1.6 GHz, 7.3 GHz, 7.2 GHz, and 9.9 GHz which correspond to values very different than the normal band of frequency of the dataset (3.5-to-4.8 GHz).

4.4 Conclusions

In this Chapter, the training phase of the PVT Estimator' ANNs for the optimization defined in Chapter 3 was described, step by step. First the datasets for each ANN were defined along with the necessary pre-processing to obtain the best data for their training phase. Furthermore, the tuning phase of the hyperparameters of the ANNs was described followed by the showcase of the final model structure. Finally, the test results of one of the ANNs were shown. These results show that the ANN is capable of achieving highly accurate results on unseen data, and therefore, are considered adequate for the remaining experiments conducted on this dissertation.

Table 4.12 - Error results of oscillation frequency for corner FF in mode b_{0000}

best			25th			median			75th			worst		
mape	real (GHz)	pred (GHz)	mape	real (GHz)	pred (GHz)	mape	real (GHz)	pred (GHz)	mape	real (GHz)	pred (GHz)	mape	real (GHz)	pred (GHz)
0.000001	4.863	4.863	0.075278	6.078	5.940	0.162609	4.792	4.765	0.305878	5.282	5.211	4.944724	4.155	4.891
0.000068	4.363	4.363	0.075308	4.508	4.522	0.162613	4.346	4.325	0.305967	4.923	4.898	5.264267	7.322	6.371
0.000082	4.814	4.814	0.075328	5.175	5.147	0.162647	3.137	3.091	0.305973	4.751	4.754	6.010913	7.184	6.107
0.000098	4.632	4.632	0.075337	5.353	5.278	0.162688	6.054	5.957	0.305991	5.047	5.028	6.157325	1.619	2.380
0.000104	2.535	2.535	0.075396	5.554	5.788	0.162689	5.356	5.342	0.306057	4.987	4.979	6.507957	9.852	8.512

Table 4.13 - Error results of power for corner FF in mode b_{0000}

best			25th			median			75th			worst		
mape	real (mW)	pred (mW)	mape	real (mW)	pred (mW)	mape	real (mW)	pred (mW)	mape	real (mW)	pred (mW)	mape	real (mW)	pred (mW)
0.000024	1.4359	1.4359	0.074626	1.1031	1.0881	0.162755	1.8863	1.8348	0.313062	1.2639	1.2621	9.834139	1.242	2.2322
0.000065	2.6972	2.6972	0.074632	1.4189	1.4346	0.162789	2.0585	2.1066	0.313096	2.1122	2.0964	9.920227	1.1652	2.1565
0.000115	1.2918	1.2918	0.074652	1.0335	1.0405	0.162803	3.124	2.868	0.313120	1.7466	1.722	11.163885	1.1188	2.2292
0.000148	1.4345	1.4345	0.074655	1.0408	1.0333	0.162855	1.1989	1.1872	0.313153	1.5612	1.5617	12.081931	0.9818	2.1669
0.000171	2.4626	2.4626	0.074668	1.3207	1.3805	0.162883	1.7028	1.6609	0.313230	1.5466	1.5332	13.018456	2.1257	3.5516

Table 4.14 - Error results of phase noise in 10MHz for corner FF in mode b_{0000}

best			25th			median			75th			worst		
mape	real	pred	mape	real	pred	mape	real	pred	mape	real	pred	mape	real	pred
0.000119	-133.73	-133.73	0.153637	-132.45	-132.48	0.279812	-134.01	-133.90	0.421004	-131.99	-131.88	11.663282	-133.41	-128.99
0.000203	-134.14	-134.14	0.153716	-133.30	-133.08	0.279954	-134.42	-134.31	0.421062	-131.29	-131.18	11.963589	-132.68	-128.06
0.000247	-132.22	-132.22	0.154076	-132.37	-132.17	0.280005	-134.16	-133.25	0.421202	-134.43	-134.25	13.839280	-135.65	-130.71
0.000250	-132.14	-132.14	0.154117	-132.02	-132.01	0.280009	-129.20	-130.29	0.421402	-134.41	-134.31	14.683453	-124.36	-131.26
0.000482	-132.69	-132.69	0.154148	-129.12	-129.05	0.280018	-133.54	-133.33	0.421456	-133.24	-133.11	16.450449	-130.22	-136.98

Table 4.15 - Error results of FOM in 10MHz for corner FF in mode b_{0000}

best			25th			median			75th			worst		
mape	real	pred	mape	real	pred	mape	real	pred	mape	real	pred	mape	real	pred
0.000051	183.41	183.41	0.070950	187.70	187.62	0.136466	184.86	184.83	0.216889	185.43	185.22	9.023088	162.52	167.61
0.000078	183.44	183.44	0.070956	184.86	184.64	0.136569	184.05	183.84	0.217096	181.88	181.80	12.927639	165.59	173.27
0.000137	185.86	185.86	0.070989	186.50	186.30	0.136571	179.14	177.82	0.217208	185.54	185.42	13.209411	151.36	157.33
0.000217	185.13	185.13	0.071010	186.42	186.39	0.136578	184.05	184.72	0.217265	186.54	186.44	14.978211	164.17	172.86
0.000240	186.17	186.17	0.071055	182.80	182.87	0.136634	185.81	185.64	0.217528	185.31	185.23	17.360127	163.12	173.01

5 AIDA Integration and Results

In this Chapter a clear explanation of the integration in the AIDA loop of the models developed in Chapter 4 will be presented and, the results of three different and optimizations using the modified AIDA loop will be shown and discussed.

5.1 AIDA Integration

With the ANNs for each corner and tuning mode tuned and ready to be used, the next step of this work is to integrate the PVT estimator into the AIDA loop. The location of the PVT estimator will follow the proposed solution in Chapter 3 and is presented in Fig. 5.1.

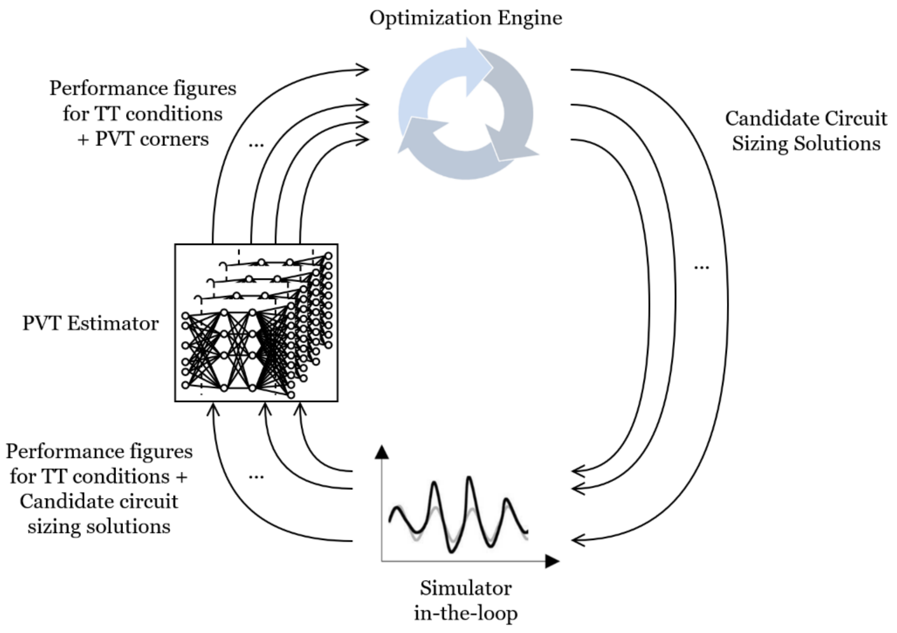

Optimization Engine
Performance figures for TT conditions + PVT corners
...
Candidate Circuit Sizing Solutions
...
PVT Estimator
Performance figures for TT conditions + Candidate circuit sizing solutions
...
Simulator in-the-loop

Figure 5.1 - Location of the PVT estimator in the AIDA optimization loop

In the first step of the optimization loop several circuit sizing solutions are proposed by the optimization engine (number of solutions depends on the population defined). In the original AIDA loop, the simulator evaluates each of these solutions for all TT conditions and PVT corners, and outputs all the evaluated performances. The optimization engine receives these evaluations and ranks the solutions (population) according to their compliance with the objectives and constraints set for the current optimization problem. A brief flowchart of one generation of the original AIDA loop is shown in Fig. 5.2.

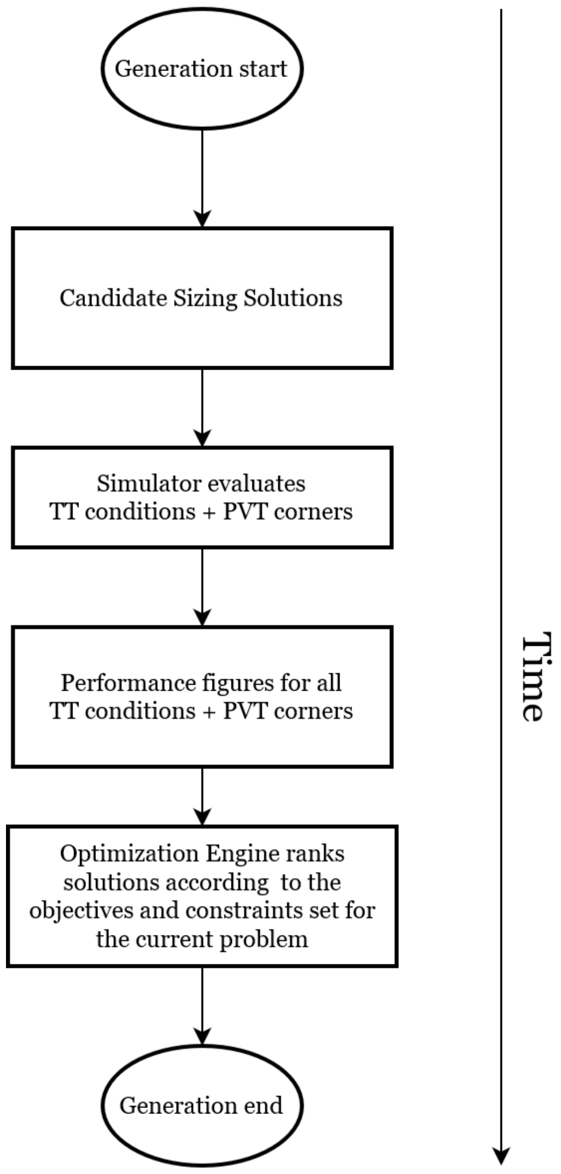

Generation start
Candidate Sizing Solutions
Simulator evaluates TT conditions + PVT corners
Performance figures for all TT conditions + PVT corners
Optimization Engine ranks solutions according to the objectives and constraints set for the current problem
Generation end
Time

Figure 5.2 - Flow of a generation of the original AIDA loop

5.1.1 Simulation-based sizing with PVT estimator

In the modified AIDA loop, the simulator will only need to evaluate the solutions for TT conditions. Each of the ANNs will receive, as input, the sizing solution and, according to its tuning mode, the performance figures respective to the TT conditions (previously evaluated by the simulator). With the inputs defined, each ANN will output the performance figures corresponding to a specific corner and tuning mode, so the performance figures for all PVT corners can be sent to the optimization engine for further ranking. These performance figures are a mix of simulated performance figures (TT corners) and predicted performance figures (remaining PVT corners).

In the scope of the sizing optimization, one of these loops represents a generation of the optimizer. A brief flowchart of one generation of the modified loop is shown in Fig. 5.3.

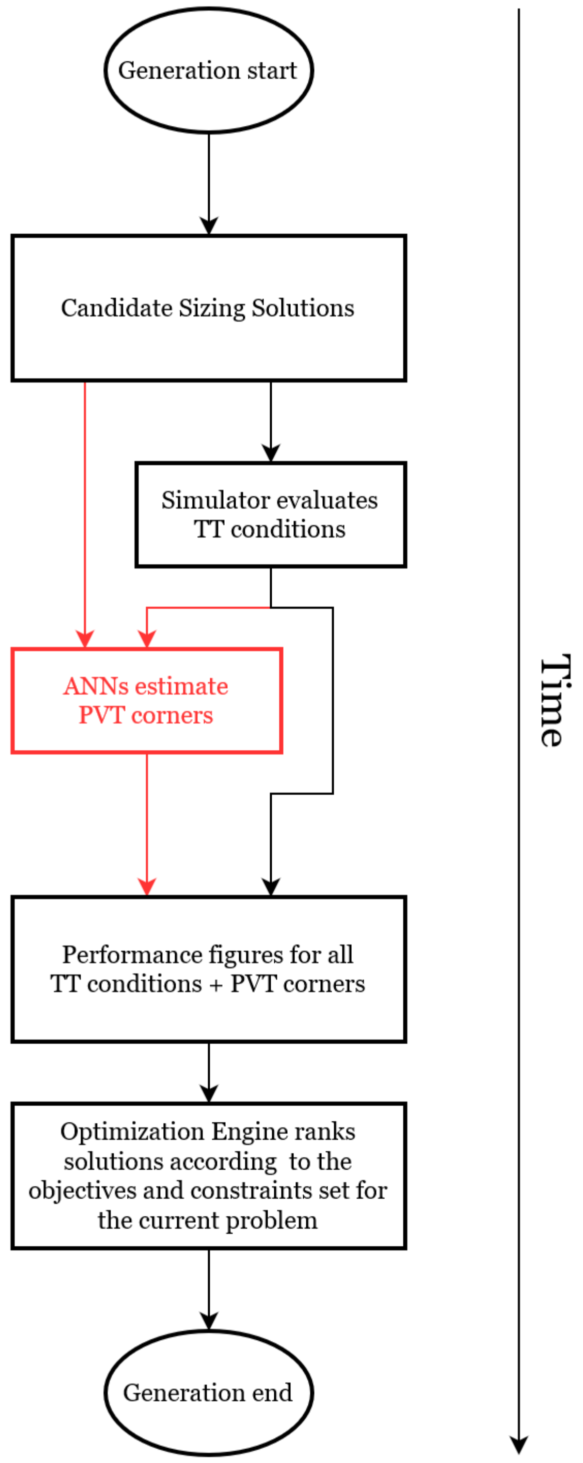

Generation start
Candidate Sizing Solutions
Simulator evaluates TT conditions
ANNs estimate PVT corners
Performance figures for all TT conditions + PVT corners
Optimization Engine ranks solutions according to the objectives and constraints set for the current problem
Generation end
Time

Figure 5.3 - Flow of a generation of the modified AIDA loop (modification in red)

The structure of the PVT estimator is further detailed in Fig. 5.4.

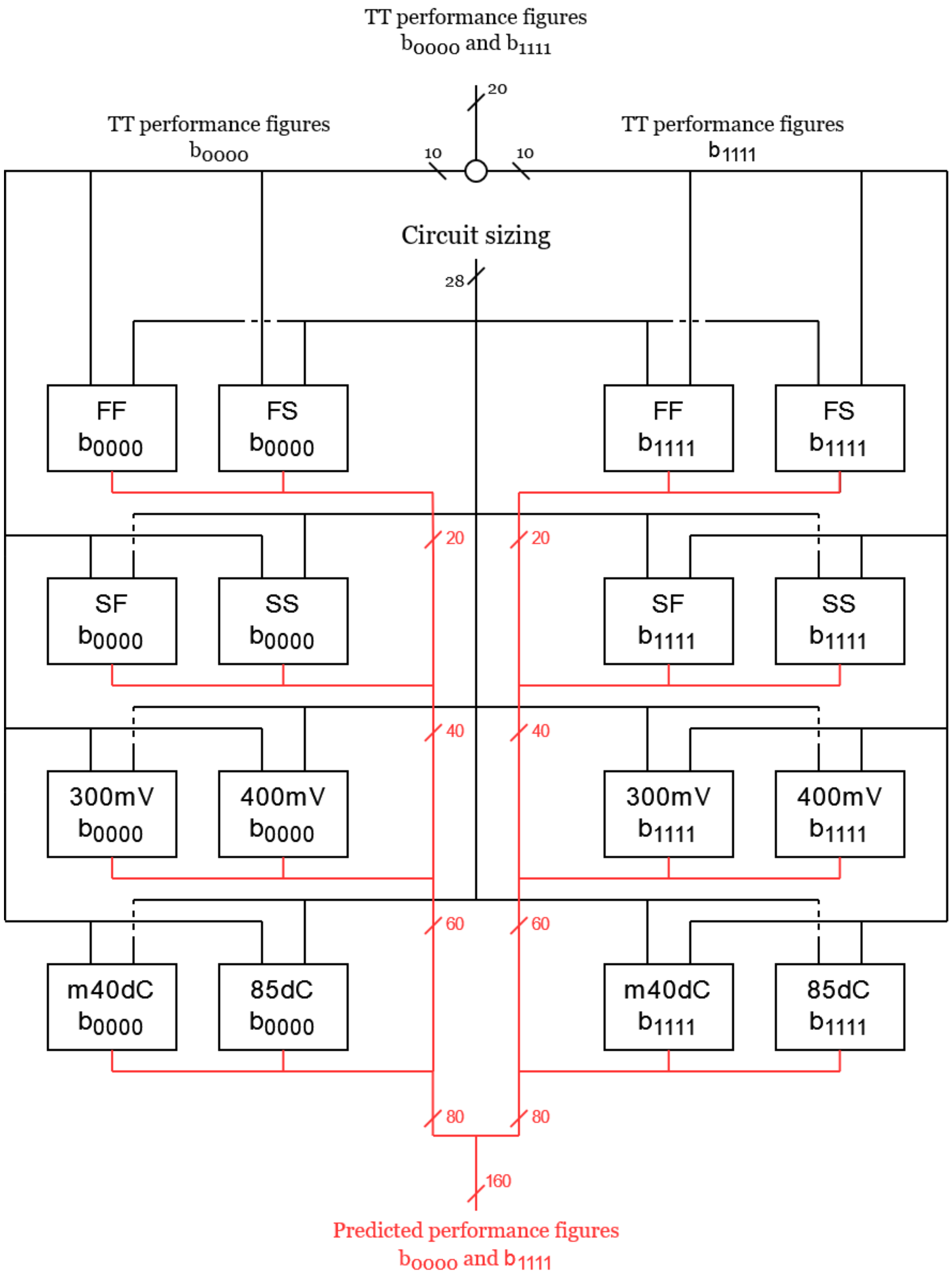


Figure 5.4 - Detailed diagram of the PVT estimator

In the following sections, a comparison with an exhaustive circuit sizing optimization is performed to determine if the optimization with the modified loop, while using the PVT estimator throughout 100% of the optimization, achieves adequate results.

5.1.2 Class C/D VCO optimization without PVT estimator

The optimization that will be utilized to compare results is the one performed to obtain the dataset used to train the ANNs (as described in Section 3.2). This comparison will serve as a primary filter to observe if the modified loop is capable of achieving feasible solutions or improving the original optimization results, while simultaneously accelerating the optimization with a speed-up factor of 9. This speed-up factor results from the optimization with the modified loop only evaluating the TT conditions for two tuning modes and the original optimization which evaluated both TT conditions and PVT corners for 2 different tuning modes ($\frac{18}{2} = 9$).

The principal objective of this optimization was to minimize both power and phase noise at 10 MHz in both tuning modes while imposing value constraints on 7 measured performances, in both tuning modes as well. These optimization constraints and objectives are shown in Table 5.1. The optimization was executed through a total of 350 generations, took a total of 612 hours to complete and resulted in a total of 27 sizing solutions. In Fig. 5.5 is shown the POF evolution of the original optimization which contains the best sizing solutions throughout the generations, and, in Table 5.2 the values of the final POF obtained at generation 350 are shown.

Table 5.1 - Optimization constraints and objectives

Tuning mode	Measure	Testbenches	Units	Optimization Constraint	Optimization Objective
b_{0000}	f_{osc}	All	GHz	≥ 4.8	
	PN@10kHz	All	dBc/Hz	≤ -49	
	PN@100kHz	All	dBc/Hz	≤ -76	
	PN@1MHz	All	dBc/Hz	≤ -98	
	PN@10MHz	All	dBc/Hz	≤ -119	minimize
	power	All	mW	n/d	minimize
	FOM@10MHz	All	dBc/Hz	≥ 180	
b_{1111}	f_{osc}	All	GHz	≤ 3.9	
	PN@10kHz	All	dBc/Hz	≤ -55	
	PN@100kHz	All	dBc/Hz	≤ -82	
	PN@1MHz	All	dBc/Hz	≤ -103	
	PN@10MHz	All	dBc/Hz	≤ -124	minimize
	power	All	mW	n/d	minimize
	FOM@10MHz	All	dBc/Hz	≥ 180	

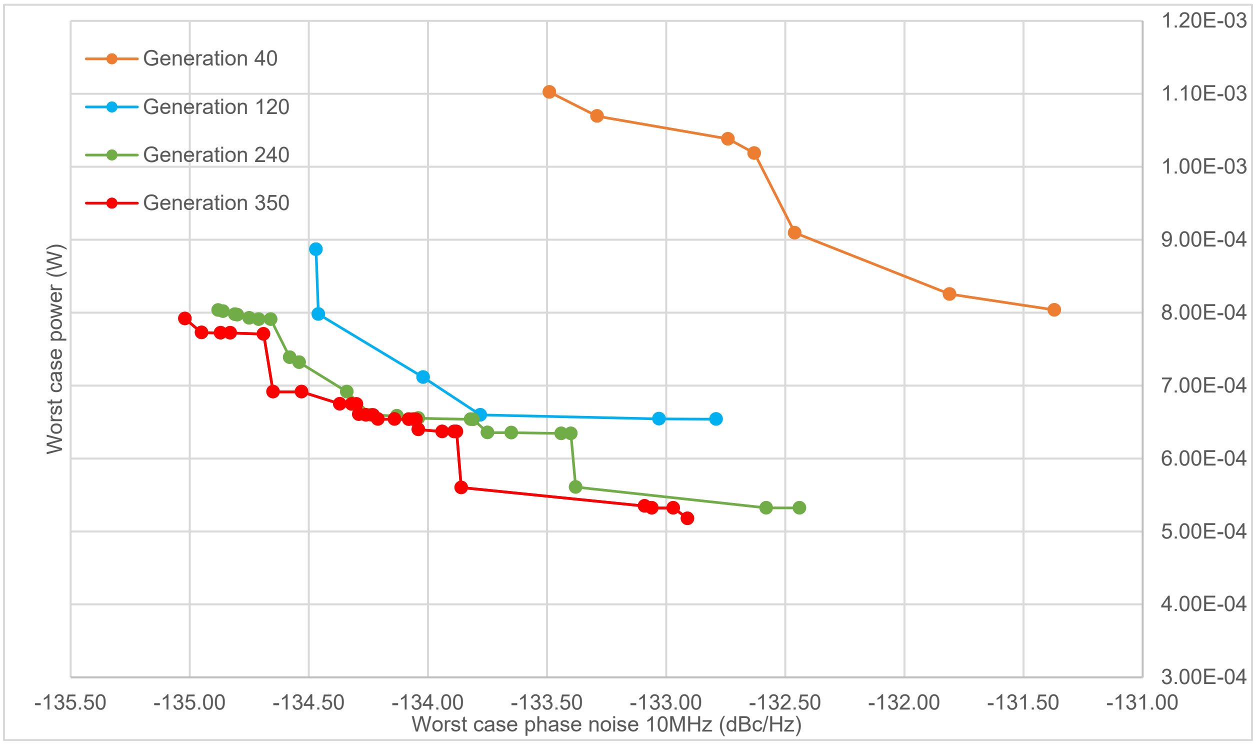


Figure 5.5 - POF evolution throughout the original optimization

Table 5.2 - Final solutions of the original POF at generation 350

Worst case Phase Noise 10MHz (dBc/Hz)	Worst case Power (W)	Worst case Phase Noise 10MHz (dBc/Hz)	Worst case Power (W)
-135.02	7.9176E-04	-134.14	6.5391E-04
-134.95	7.7265E-04	-134.08	6.5374E-04
-134.87	7.7217E-04	-134.06	6.5373E-04
-134.83	7.7216E-04	-134.05	6.5360E-04
-134.69	7.7067E-04	-134.04	6.3966E-04
-134.65	6.9143E-04	-133.94	6.3690E-04
-134.53	6.9142E-04	-133.89	6.3685E-04
-134.37	6.7491E-04	-133.88	6.3679E-04
-134.32	6.7460E-04	-133.86	5.6012E-04
-134.30	6.7454E-04	-133.09	5.3484E-04
-134.29	6.6054E-04	-133.06	5.3225E-04
-134.26	6.5976E-04	-132.97	5.3219E-04
-134.23	6.5959E-04	-132.91	5.1784E-04
-134.21	6.5394E-04		

5.1.3 Class C/D VCO with PVT estimator working at 100%

To obtain the best base of comparison possible, the number of generations of the now modified optimization was set to 350 as well, the same as the original, and, both optimization constraints and objectives were the same, shown in Table 5.1. The POF evolution throughout the modified loop optimization is depicted in Fig. 5.6 alongside the final original POF.

However, in order to ascertain if the solutions obtained are feasible or not all the circuit sizing solutions must be evaluated by the simulator. Unfortunately, all the solutions revealed to be unfeasible in at least 30 of the 160 total performances. The POF values obtained in generation 350 and number of failed optimization constraints is shown in Table 5.3.

As observable, the number of final solutions obtained increased substantially (from 27 to 40) and the range of values for both worst-case power and worst-case phase noise at 10MHz improved significantly, with solutions where the worst-case power achieves 2 mW, less than half of the lowest worst-case power of the original optimization, and also, the worst-case phase noise of all solutions decreased by at least 2 dBc/Hz. However, these results cannot be possibly translated into a working circuit.

Finally, it was performed a small optimization of 20 generations with the original AIDA loop using the last state of the modified loop optimization as starting point, in order to confirm if feasible solutions can be found. However, even with this step no feasible solution was found.

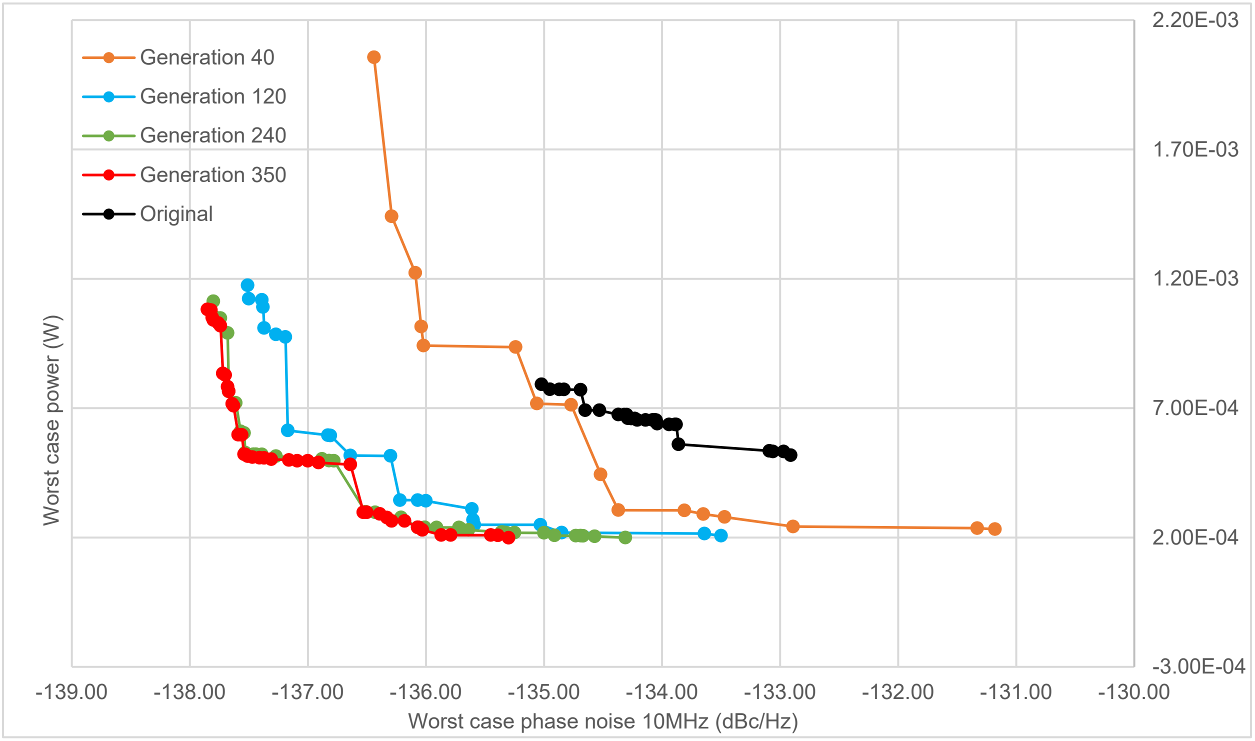


Figure 5.6 - POF evolution throughout the modified loop optimization

Table 5.3 - POF at generation 350

Worst case Phase Noise 10MHz (dBc/Hz)	Worst case Power	# Failed constraints	Worst case Phase Noise 10MHz (dBc/Hz)	Worst case Power	# Failed constraints
-137.85	1.08E-03	30	-137.31	5.02E-04	49
-137.82	1.08E-03	31	-137.16	4.99E-04	49
-137.81	1.05E-03	36	-137.09	4.96E-04	49
-137.80	1.04E-03	35	-137.00	4.96E-04	49
-137.76	1.03E-03	36	-136.91	4.89E-04	49
-137.74	1.02E-03	42	-136.64	4.82E-04	49
-137.72	8.34E-04	36	-136.53	2.97E-04	51
-137.70	8.28E-04	36	-136.50	2.97E-04	50
-137.68	7.82E-04	35	-136.39	2.91E-04	49
-137.67	7.65E-04	36	-136.33	2.77E-04	49
-137.64	7.17E-04	36	-136.29	2.64E-04	54
-137.63	7.09E-04	41	-136.18	2.64E-04	48
-137.59	5.97E-04	44	-136.07	2.39E-04	51
-137.56	5.97E-04	36	-136.06	2.39E-04	50
-137.54	5.21E-04	49	-136.03	2.29E-04	48
-137.53	5.21E-04	49	-135.87	2.10E-04	49
-137.51	5.14E-04	49	-135.79	2.09E-04	51
-137.47	5.10E-04	49	-135.45	2.09E-04	49
-137.41	5.08E-04	49	-135.39	2.08E-04	55
-137.37	5.08E-04	49	-135.30	1.99E-04	55

Conclusion

These results demonstrate one major flaw in the modified loop, it over-estimates the circuit performances, i.e., it makes the optimization engine search in a performance space where the circuit simply cannot operate in real world conditions. In order to prevent this situation, some control has to be introduced in the modified loop to guide the optimization into feasible regions.

5.1.4 PVT estimator with error controller

As stated, some control has to be introduced in the loop to better guide the optimization to feasible solution regions. In order to achieve this, a simple error controller for each ANN used in the modified loop was implemented. A brief flowchart of one generation of the modified loop with the new controller is shown in Fig. 5.7.

At each generation, before both candidate sizing solutions and TT performance figures are sent to the ANNs, first they pass through a controller that will choose which ANNs will operate at the current generation, i.e., which PVT corner/tuning mode combination will be simulated or predicted.

First, the controller sends 20% of the candidate sizing solutions to be simulated and predicted at the same time. For PVT corner/tuning mode combination, with the output of the simulator, the controller checks if there are more feasible solutions than unfeasible solutions. If there are more unfeasible solutions than feasible solutions or the same number, the PVT corner/tuning mode combination will be simulated (instead of predicted) in that generation for the remaining candidate solutions. This acts as the primary filter to prevent the optimization of entering unfeasible regions.

If there are more feasible solutions than unfeasible it passes to the next step. Here the controller uses the output of the feasible solutions from the simulator, i.e., simulated performances (from the previous step) and compares them to the corresponding predicted performances. The error between each performance is calculated and the average error of all points is obtained. If the error (MAPE) is higher than 5% the combination will be simulated in that generation for the rest of the candidate solutions. If is equal or lower than 5% the corresponding ANN will predict the PVT corner/tuning mode combination in that generation for the rest of the candidate solutions.

The flow of the controller is depicted in Fig. 5.8.

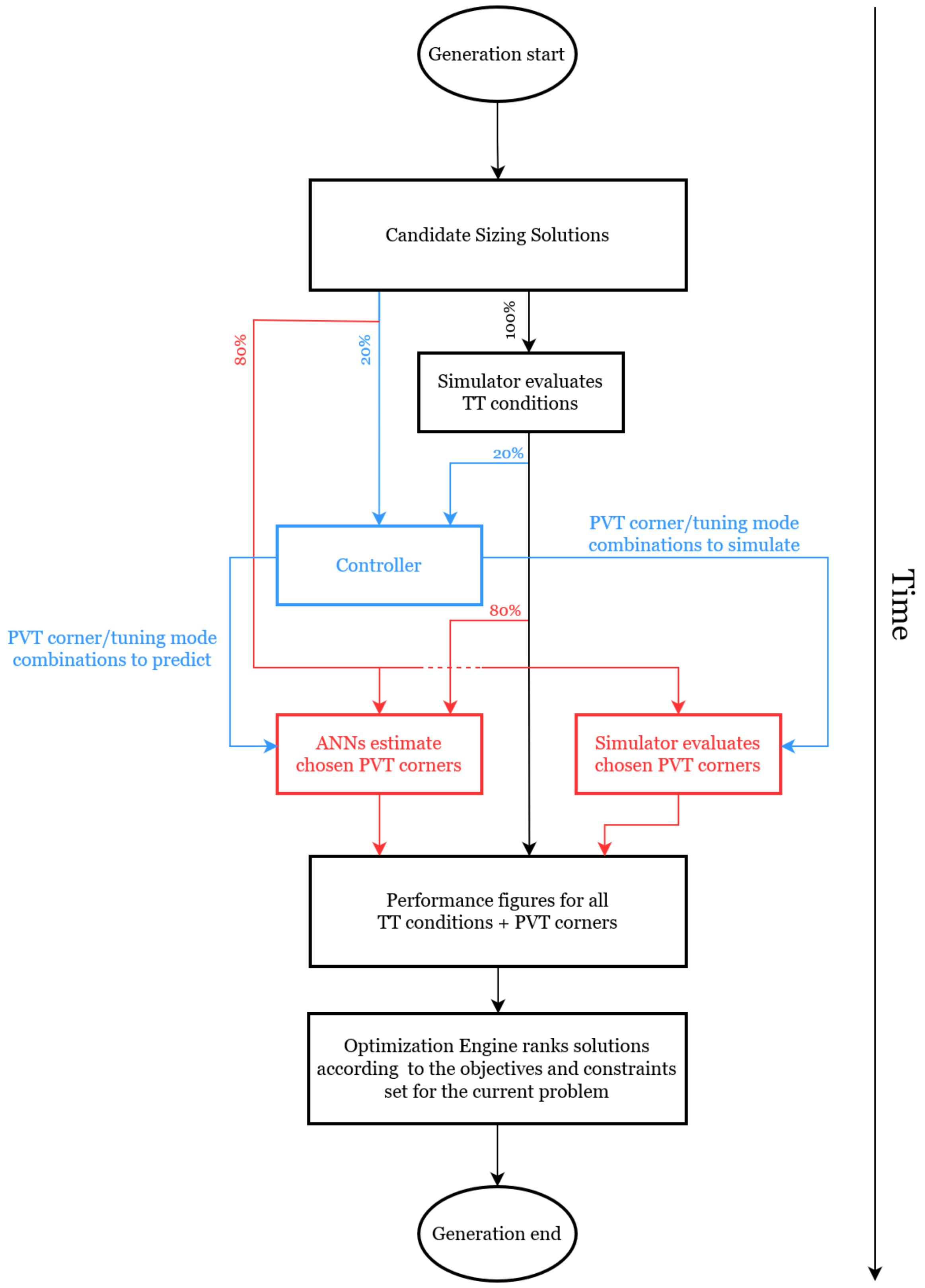

Generation start
Candidate Sizing Solutions
100%
80%
20%
Simulator evaluates TT conditions
20%
Controller
PVT corner/tuning mode combinations to simulate
80%
PVT corner/tuning mode combinations to predict
ANNs estimate chosen PVT corners
Simulator evaluates chosen PVT corners
Performance figures for all TT conditions + PVT corners
Optimization Engine ranks solutions according to the objectives and constraints set for the current problem
Generation end
Time

Figure 5.7 - Flow of a generation of the controlled modified AIDA loop (modification in red and controller in blue)

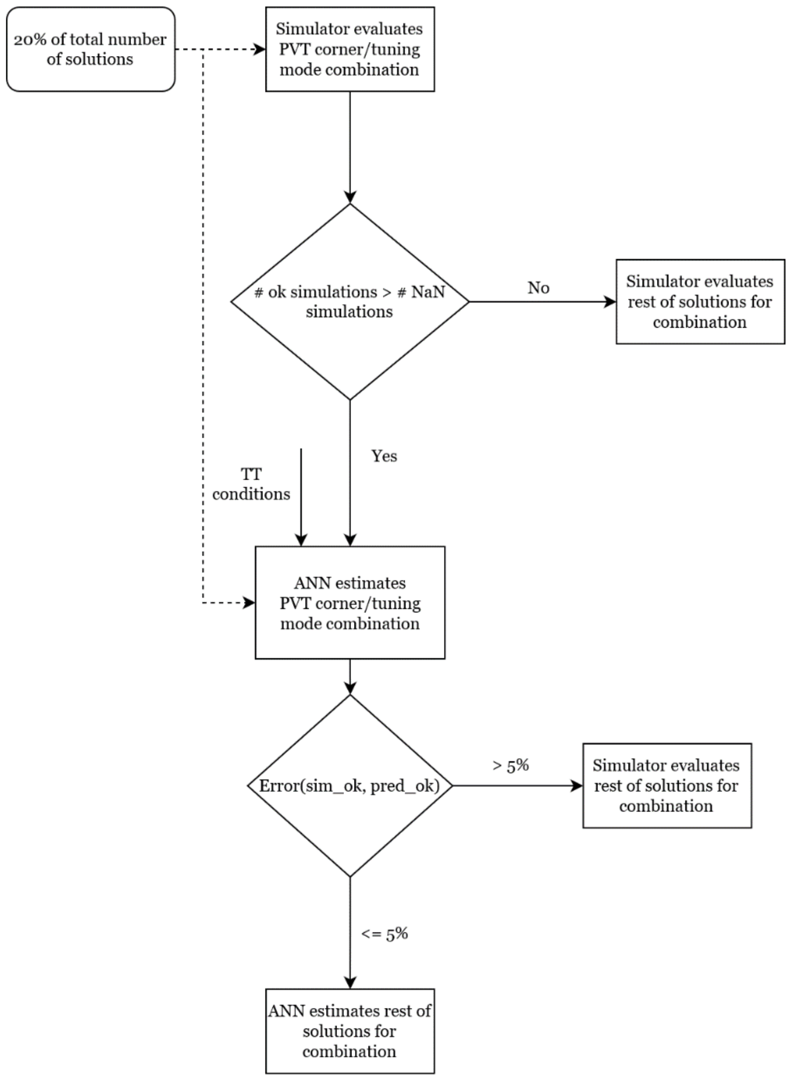

20% of total number of solutions
Simulator evaluates PVT corner/tuning mode combination
# ok simulations > # NaN simulations
No
Simulator evaluates rest of solutions for combination
Yes
TT conditions
ANN estimates PVT corner/tuning mode combination
Error(sim_ok, pred_ok)
> 5%
Simulator evaluates rest of solutions for combination
<= 5%
ANN estimates rest of solutions for combination

Figure 5.8 - Flow of the controller for each PVT corner/tuning mode combination

5.2 Results with controlled PVT estimator

Using the new controlled PVT estimator, three different optimizations will be performed, where in each one a different objective of this dissertation will be analyzed:

1. **Class C/D VCO for 3.5-to-4.8 GHz:** check if the controlled PVT estimator is capable of adequate circuit performance estimations in the PVT corners;
2. **Class C/D VCO 2.3 GHz-to-2.5 GHz:** verify if the controlled PVT estimator can be directly reused for an optimization with completely different objectives and constraints of the same topology that was trained for, i.e., without re-training it (plug-and-play functionalities);
3. **ULP Class B/C VCO:** verify if the same ANN structure used in the controlled PVT estimator for a particular circuit topology can be reused for a different VCO circuit topology (plug-and-train functionalities).

5.2.1 Class C/D VCO for 3.5-to-4.8 GHz

In this section the main objective is to compare the final POF results using the controlled PVT estimator with the final POF obtained in the original optimization in Fig. 5.5. The number of generations of the optimization using the controlled PVT estimator was set to 330 and in the last 20 generations the optimization was carried using the original loop shown in Fig. 5.2. The POF evolution throughout the optimization with the controlled PVT estimator is depicted in Fig. 5.9 alongside the final original POF.

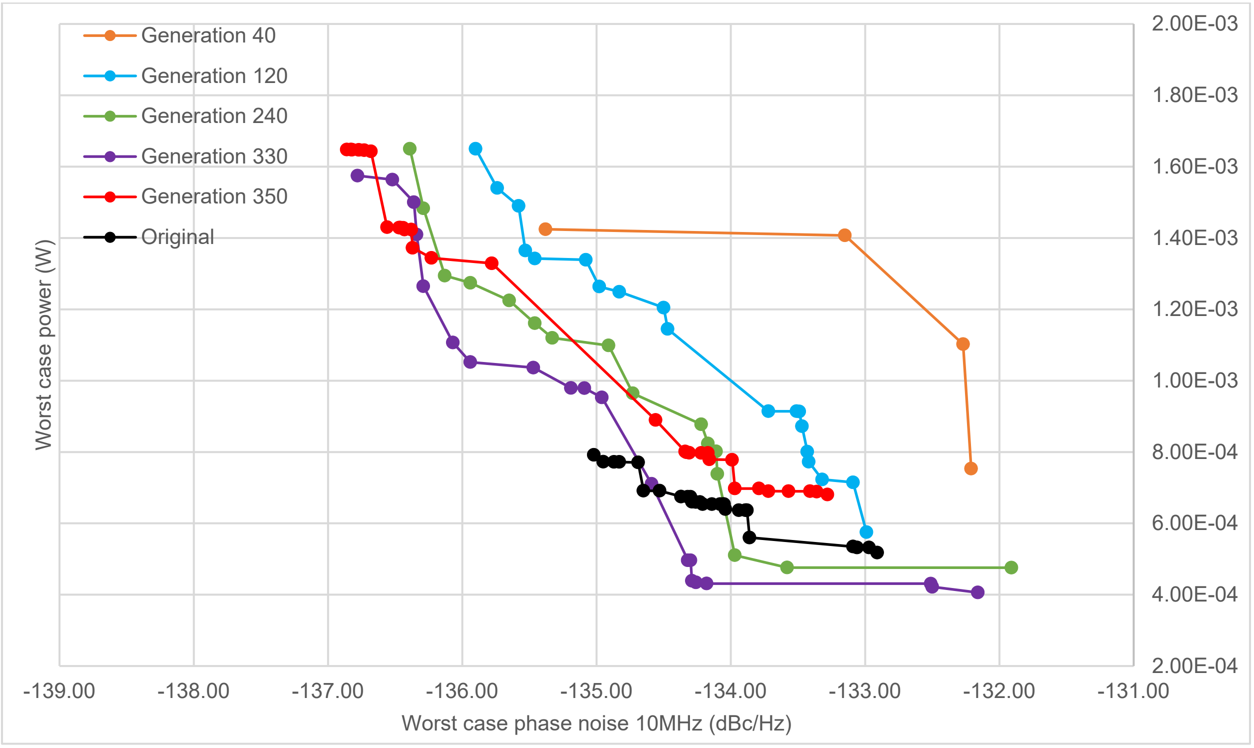


Figure 5.9 - POF evolution throughout the modified loop optimization with controller

As observable, the optimization at generation 330, i.e., the optimization using the controlled PVT estimator finds solutions with similar worst case power and worst case phase noise at 10 MHz to the original optimization when comparing with the optimization using a PVT estimator working at 100% in Fig. 5.6.

The circuit sizing solutions of the POF at generation 330 were simulated and the results show that 1 of the 20 solutions of the POF passed all the optimization constraints and thus feasible solutions were found. Additionally, 2 solutions close to the optimization specifications were also found, with only 1 constraint failed. These results are shown in better detail in Table 5.4. This concludes that the controller was capable of directing the optimization to feasible performance regions, which was the main problem of the optimization using the PVT estimator working at 100%.

Table 5.4 - POF at generation 330

Worst case Phase Noise 10MHz (dBc/Hz)	Worst case Power (W)	# Failed Constraints	Worst case Phase Noise 10MHz (dBc/Hz)	Worst case Power (W)	# Failed Constraints
-136.78	1.57E-03	1	-134.96	9.53E-04	24
-136.52	1.56E-03	13	-134.59	7.11E-04	23
-136.36	1.50E-03	0	-134.32	4.96E-04	30
-136.34	1.41E-03	1	-134.30	4.96E-04	30
-136.29	1.26E-03	7	-134.29	4.39E-04	30
-136.07	1.11E-03	13	-134.26	4.35E-04	30
-135.94	1.05E-03	18	-134.18	4.31E-04	30
-135.47	1.04E-03	13	-132.51	4.31E-04	47
-135.19	9.79E-04	18	-132.50	4.22E-04	30
-135.09	9.79E-04	13	-132.16	4.06E-04	36

After the last 20 generations, the optimization was capable of finding 30 solutions, and because these generations were performed fully with the simulator, all of them are feasible. Comparing the performances of these solutions with the original POF, it is possible to observe that the solutions in terms of worst case power are worse than the original POF by at least 1 mW but regarding the worst case phase noise at 10 MHz the optimization was capable of finding solutions with nearly less 2 dBc/Hz. The final POF is shown in Fig. 5.9 and the values are shown in better detail in Table 5.5.

Table 5.5 - POF at generation 350

Worst case Phase Noise 10MHz (dBc/Hz)	Worst case Power (W)	Worst case Phase Noise 10MHz (dBc/Hz)	Worst case Power (W)	Worst case Phase Noise 10MHz (dBc/Hz)	Worst case Power (W)
-136.86	1.65E-03	-136.43	1.42E-03	-134.17	7.97E-04
-136.83	1.65E-03	-136.38	1.42E-03	-134.16	7.79E-04
-136.82	1.65E-03	-136.37	1.37E-03	-133.99	7.78E-04
-136.77	1.65E-03	-136.23	1.34E-03	-133.97	6.98E-04
-136.73	1.65E-03	-135.78	1.33E-03	-133.79	6.98E-04
-136.68	1.64E-03	-134.56	8.90E-04	-133.72	6.90E-04
-136.56	1.43E-03	-134.34	8.01E-04	-133.57	6.90E-04
-136.47	1.43E-03	-134.33	7.99E-04	-133.41	6.90E-04
-136.46	1.43E-03	-134.31	7.98E-04	-133.36	6.89E-04
-136.44	1.43E-03	-134.22	7.98E-04	-133.28	6.81E-04

The speed-up obtained with the controlled PVT estimator is calculated by using the percentage of usage of each ANN throughout the first 330 generation of the optimization. These values are shown in Fig. 5.10.

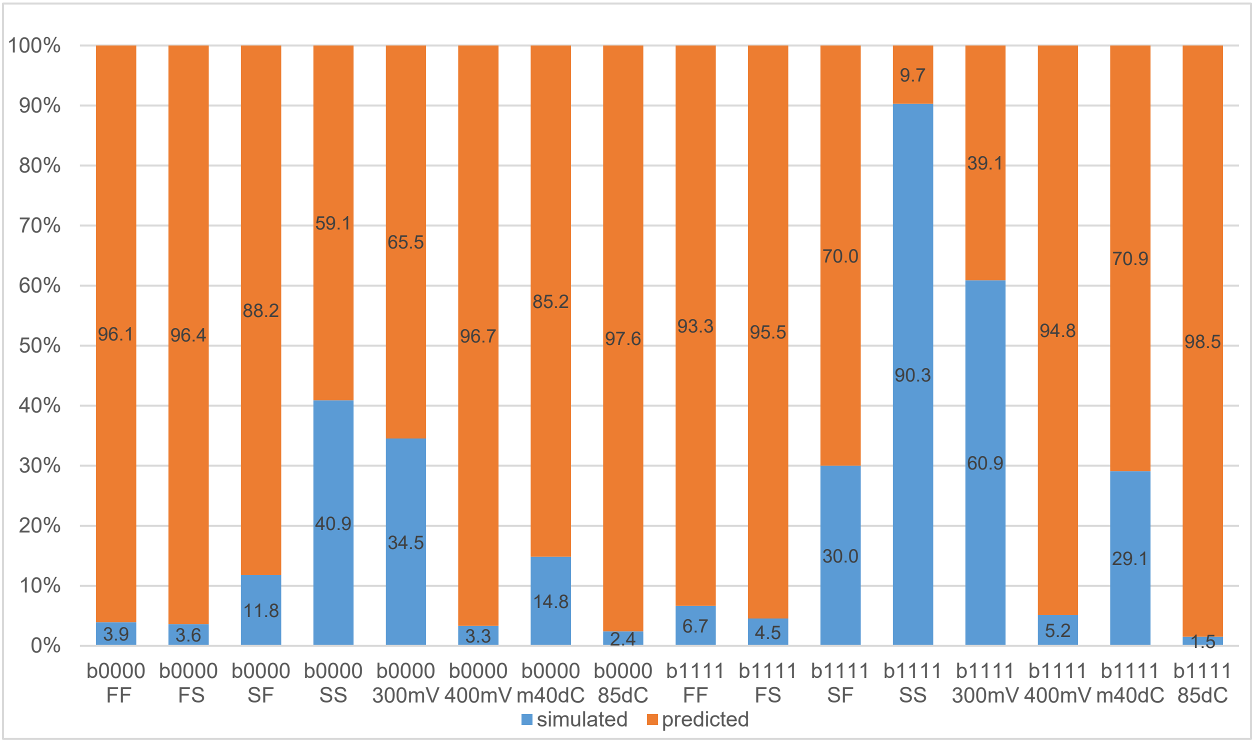


Figure 5.10 - Modes simulated vs predicted throughout optimization

While analyzing Fig. 5.10, it is possible to report that throughout the first 330 generations of the optimization 78.5% of the original PVT corners evaluations were instead predicted using ANNs. Considering the simulation of the TT conditions, using the controlled PVT estimator resulted in a speed-up factor of 3.31.

However, the total speed-up of the 350 generations is lower due to the last 20 generations of full simulation that must be accounted. Considering these simulations, the total speed-up factor obtained using the controlled PVT estimator is 2.92.

Conclusion

These results show that the optimization using a controlled PVT estimator is capable of finding a small group of feasible solutions or almost feasible solutions capable of competing in terms of performance with the original optimization while achieving a speed-up factor of 3.31. In the original optimization, the first 330 generations took, approximately, 577 hours to complete, while the optimization using the PVT estimator only took 185 hours to complete. Using the simulator for an additional 20 generations at the end of the optimization achieves a more robust and broad-ranging POF, while still capable of achieving a speed-up factor of 2.92. With the PVT estimator, in total, the modified optimization took 16 and a half days less than the original optimization. The main objective of this comparison is therefore confirmed to be possible, proving that the PVT estimator is capable of finding adequate PVT corner circuit performance estimations.

5.2.2 Plug-and-play Class C/D VCO 2.3 GHz-to-2.5 GHz

The main objective in this section is to determine if the controlled PVT estimator, trained and tuned for the optimization of Section 5.2.1, is capable of finding adequate PVT corner circuit performances estimations for an optimization with completely different targets of the same circuit topology. For the comparison, the optimization that will be used is a similar simulation-based sizing optimization of the circuit in Section 3.2 but the range of the oscillation frequency in which the class C/D VCO will operate is now set to 2.3 GHz-to-2.5 GHz and the optimization constraints were tighten, i.e., the maximum (or minimum) values chosen for the measured performances were decreased (or increased). The optimization constraints and objectives are shown in Table 5.6.

Table 5.6 - Optimization constraints and objectives

Tuning mode	Measure	Testbenches	Units	Optimization Constraint	Optimization Objective
b_{0000}	f_{osc}	All	GHz	≥ 2.5	
	PN@10kHz	All	dBc/Hz	≤ -54	
	PN@100kHz	All	dBc/Hz	≤ -81	
	PN@1MHz	All	dBc/Hz	≤ -103	
	PN@10MHz	All	dBc/Hz	≤ -124	minimize
	power	All	mW	n/d	minimize
	FOM@10MHz	All	dBc/Hz	≥ 185	
b_{1111}	f_{osc}	All	GHz	≤ 2.3	
	PN@10kHz	All	dBc/Hz	≤ -60	
	PN@100kHz	All	dBc/Hz	≤ -87	
	PN@1MHz	All	dBc/Hz	≤ -108	
	PN@10MHz	All	dBc/Hz	≤ -129	minimize
	power	All	mW	n/d	minimize
	FOM@10MHz	All	dBc/Hz	≥ 185	

First an optimization using the unmodified loop was executed only for comparison terms, however, as opposed to the original optimization of Section 5.1.2, the data will not be used for the training phase of the ANNs, given that it would defeat the purpose of the main objective of this comparison. The optimization executed a total of 200 generations, took a total of 350 hours to complete and resulted in a total of 13 sizing solutions. In Fig. 5.11 is shown the POF evolution of the original optimization which contains the best sizing solutions throughout the generations and in Table 5.7 is shown the values of the final POF obtained at generation 200.

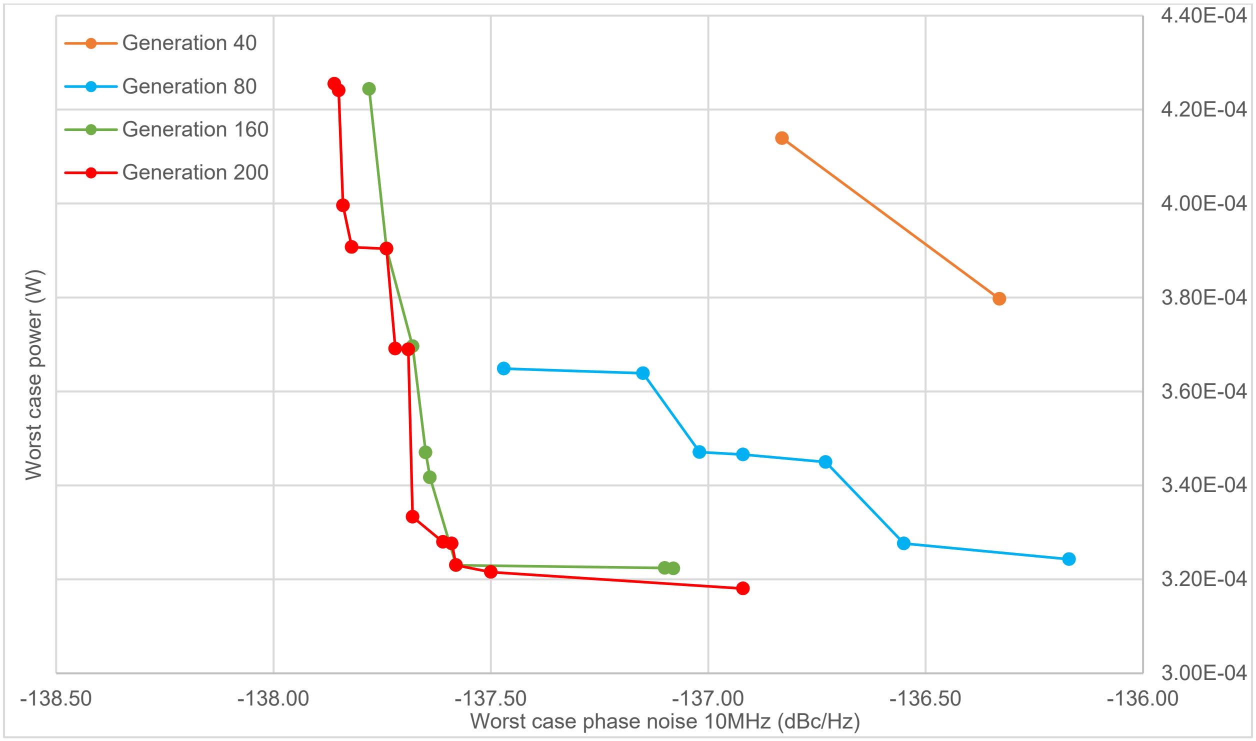


Figure 5.11 - POF evolution throughout the original optimization

Table 5.7 - Original POF at generation 200

Worst case Phase Noise 10MHz (dBc/Hz)	Worst case Power (W)
-137.86	4.25E-04
-137.85	4.24E-04
-137.84	4.00E-04
-137.82	3.91E-04
-137.74	3.90E-04
-137.72	3.69E-04
-137.69	3.69E-04
-137.68	3.33E-04
-137.61	3.28E-04
-137.59	3.28E-04
-137.58	3.23E-04
-137.50	3.22E-04
-136.92	3.18E-04

Similar to the optimization of Section 5.2.1, the optimization using the controlled PVT estimator was executed for 180 generations and, after this, the optimization was carried for 20 generations using the original unmodified loop shown in Fig. 5.2. The POF evolution throughout this optimization is depicted in Fig. 5.12 alongside the final original POF for easier comparison.

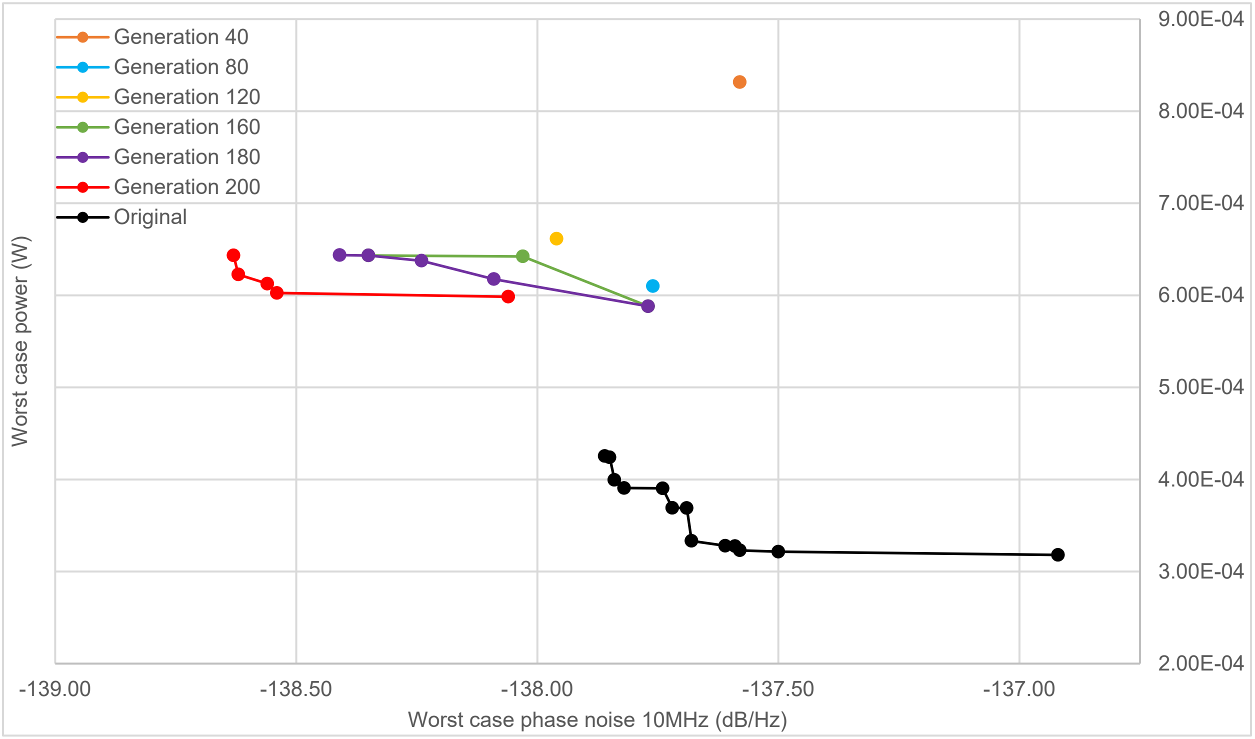


Figure 5.12 - POF evolution throughout the modified loop optimization with controller

As can be seen by the results, the optimization at generation 180 was capable of finding solutions in a performance region with better worst case phase noise at 10 MHz than the original optimization. However, the worst case power almost doubled when comparing the solutions with the lowest worst case power of both optimizations.

To ascertain if the solutions found are feasible or not and how many optimization specification constraints were failed, if any, these were simulated. The results show that no solution passed all optimization specification constraints, however 2 of them achieved only 2 failed constraints and 2 other achieved 7 failed constraints. This means that these solutions came really close to feasibility which indicates that the optimization was capable of finding an adequate POF performance region. These results are shown in Table 5.8.

Table 5.8 - POF at generation 180

Worst case Phase Noise 10MHz (dBc/Hz)	Worst case Power (W)	# Failed Constraints
-138.41	6.44E-04	7
-138.35	6.43E-04	2
-138.24	6.38E-04	2
-138.09	6.18E-04	7
-137.77	5.88E-04	12

After the last 20 generations, the optimization found 5 feasible solutions near the region of performances at generation 180. When comparing results, it is clear that the lowest worst case power solution found with PVT estimator increased by almost 3 mW which is equal to a 100% increase from the original optimization. However, the optimization was capable of finding solutions with 1 dBc/Hz less than the solutions in the original POF. These results are shown in Table 5.9.

Table 5.9 - POF at generation 200

Worst case Phase Noise 10MHz (dBc/Hz)	Worst case Power (W)
-138.63	6.43E-04
-138.62	6.23E-04
-138.56	6.13E-04
-138.54	6.03E-04
-138.06	5.98E-04

Once again, the speed-up obtained with the controlled PVT estimator is calculated by using the percentage of usage of each ANN throughout the optimization. These values are shown in Fig. 5.13.

While analyzing Fig. 5.13, it is possible to report that throughout the first 180 generations of the optimization 74.5% of the original PVT corners evaluations were instead predicted using ANNs. Considering the simulation of the TT conditions, using the controlled PVT estimator resulted in a speed-up factor of 2.95.

However, and as before, the total speed-up of the 200 generations is lower as the last 20 generations of full simulation must be accounted. Considering these simulations, the total speed-up factor obtained using the controlled PVT estimator is 2.48.

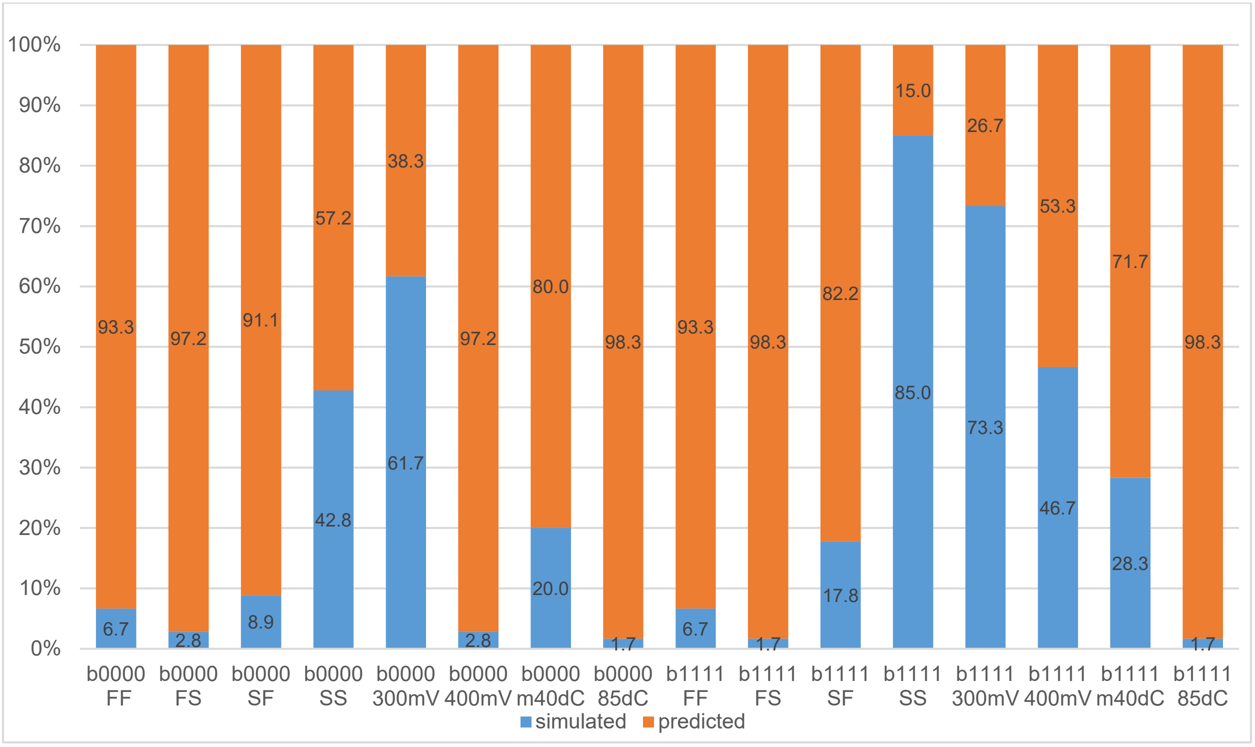


Figure 5.13 - Modes simulated vs predicted throughout optimization

Conclusion

These results show that the controlled PVT estimator is capable of finding adequate circuit sizing performance regions despite being trained and tuned with data from another optimization, while still being able to obtain a speed-up factor of 2.95. In the original optimization, the first 180 generations took, approximately, 315 hours to complete, while the optimization using the PVT estimator only took 107 hours to complete. Using the simulator for the additional 20 generations, resulted in a final POF with 5 feasible solutions while still being able to achieve a speed-up factor of 2.48. With the PVT estimator, in total, the modified optimization took 8 and a half days less than the original optimization. The final POF results obtained reveal feasible solutions with better performances in terms of phase noise but worse performances in terms of power. Due to the random nature of the optimization loop, it is not certain that these performance results would be replicated if the optimization would be executed again. To ascertain the competitiveness of these solutions in terms of performances more than, for example, 5 optimizations would have to be performed. However, the reduction of optimization days using the PVT estimator reveals the competitiveness in terms of computational time used to obtain feasible solutions.

5.2.3 Plug-and-train Ultralow-Power Class B/C VCO

The main objective in this section is to determine if the controlled PVT estimator, while reusing the ANN structure for a certain VCO circuit topology in the training phase, is capable of finding adequate PVT corner circuit performances estimations for a different VCO circuit topology. For the comparison, it will be used a simulation-based sizing optimization of an Ultralow-Power class B/C VCO, illustrated in Fig. 5.14. The full list of optimization variables is presented in Table 5.10.

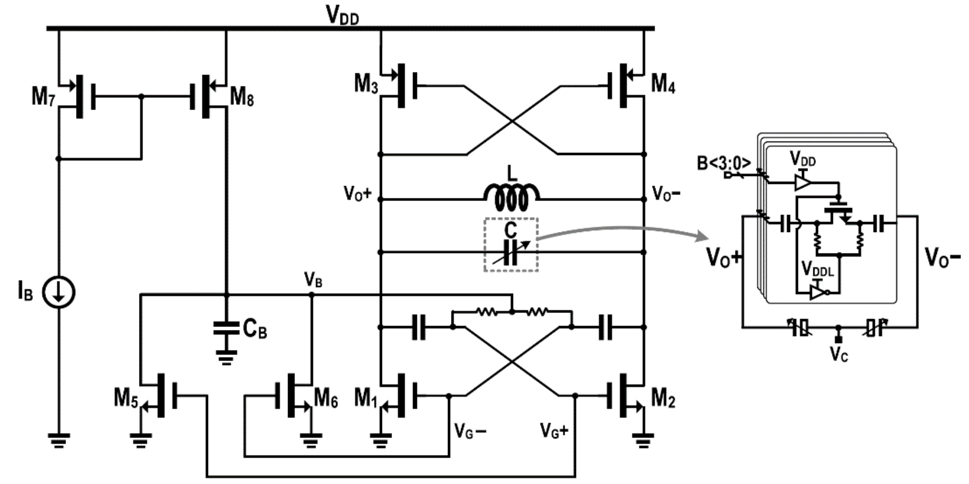


Figure 5.14 – Ultralow-Power Complementary Class B/C Hybrid-Mode VCO topology. Reprinted from [8].

Table 5.10 - Optimization variables

Variable	Units	Min.	Grid	Max.
irad	µm	15	5	90
itur	-	1	1	6
ispa	µm	2	1	4
iwid	µm	3	1	30
nccl, pccl	nm	60	20	240
nccf, pccf, $f^{5}/_{6}$, $f^{7}/_{8}$	-	1	1	32
nccw, pccw, $w^{5}/_{6}$	µm	0.6	0.2	6
$l^{7}/_{8}$	nm	130	20	6000
$w^{7}/_{8}$	nm	120	20	6000
rccl	µm	0.8	0.2	30
cnv, cnh, vnv, vnh, snv	-	6	2	100
snh	-	6	2	50

Similar to the optimization of Section 5.1.2, the principal objective was to minimize both power and phase noise at 10 MHz in both tuning modes while imposing value constraints on 7 measured performances, in both tuning modes as well. These optimization constraints are shown in Table 5.11.

Table 5.11 - Optimization constraints and objectives

Tuning mode	Measure	Testbenches	Units	Optimization Constraint	Optimization Objective
$\boldsymbol{b_{0000}}$	f_{osc}	All	GHz	≥ 5.3	
	PN@10kHz	All	dBc/Hz	≤ -55	
	PN@100kHz	All	dBc/Hz	≤ -75	
	PN@1MHz	All	dBc/Hz	≤ -95	
	PN@10MHz	All	dBc/Hz	≤ -115	minimize
	power	All	mW	n/d	minimize
	FOM@10MHz	All	dBc/Hz	≥ 175	
$\boldsymbol{b_{1111}}$	f_{osc}	All	GHz	≤ 4.6	
	PN@10kHz	All	dBc/Hz	≤ -55	
	PN@100kHz	All	dBc/Hz	≤ -75	
	PN@1MHz	All	dBc/Hz	≤ -95	
	PN@10MHz	All	dBc/Hz	≤ -115	minimize
	power	All	mW	n/d	minimize
	FOM@10MHz	All	dBc/Hz	≥ 175	

First an optimization using the unmodified loop was executed for comparison with the optimization using the PVT estimator and to obtain the dataset to train the ANNs. The data from the original optimization was collected and structured similarly to the previous dataset. To use this dataset for the training phase of the ANNs, the pre-processing strategy used was similar to the one described in Section 4.1, resulting in 16 datasets clear of outliers and null values, ready to be fed to the ANNs. The structure of the ANNs that will be used in the PVT estimator will be one tuned in Chapter 4, used in Section 5.2.1 and 5.2.2, however, as observable in Table 5.10, the number of optimization variables is now 22 which forces the number of neurons in the input layer to change from 38 to 32 (22 optimization variables plus 10 performance figures in TT conditions).

The optimization with the unmodified loop executed a total of 100 generations, took a total of 636 hours to complete and resulted in a total of 14 sizing solutions. In Fig. 5.15 is shown the POF evolution of this optimization and in Table 5.12 is shown the values of the final POF obtained at generation 100.

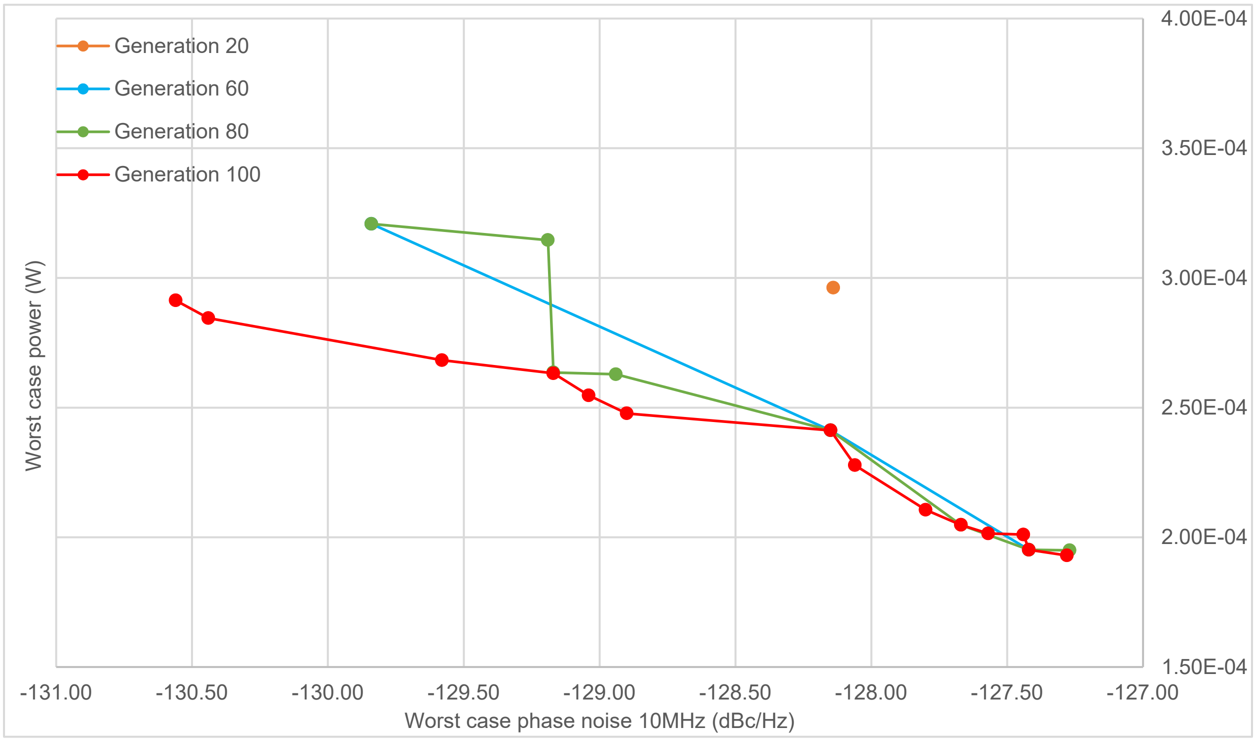


Figure 5.15 - POF evolution throughout the original optimization

Table 5.12 - Original POF at generation 100

Worst case Phase Noise 10MHz (dBc/Hz)	**Worst case Power (W)**
-130.56	2.91E-04
-130.44	2.85E-04
-129.58	2.68E-04
-129.17	2.63E-04
-129.04	2.55E-04
-128.90	2.48E-04
-128.15	2.41E-04
-128.06	2.28E-04
-127.80	2.11E-04
-127.67	2.05E-04
-127.57	2.01E-04
-127.44	2.01E-04
-127.42	1.95E-04
-127.28	1.93E-04

Using the same strategy as in the previous sections, the optimization using the controlled PVT estimator was executed for 80 generations and, after this, the optimization was carried for 20 generations using the original unmodified loop shown in Fig. 5.2. The POF evolution throughout this optimization is depicted in Fig. 5.16 alongside the final original POF for easier comparison.

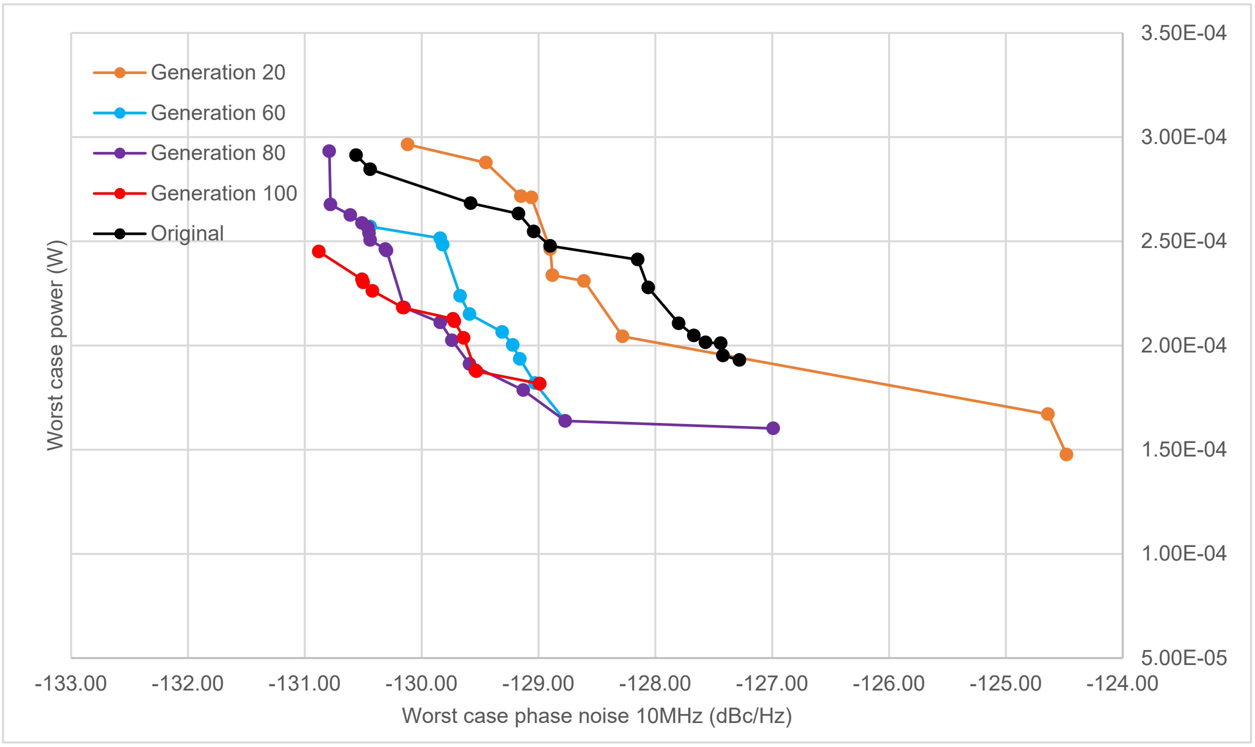


Figure 5.16 - POF evolution throughout the modified loop optimization with controller

As observable, the optimization at generation 80 was capable of finding 16 solutions in a performance region with considerable better worst case phase noise at 10 MHz than the original optimization and significantly lower worst case power throughout almost all solutions.

To ascertain if the solutions found are feasible or not and how many optimization specification constraints were failed, if any, these were simulated. The results show that no solution passed all optimization specification constraints, however 5 of them achieved 6 failed constraints and 4 other achieved 12 failed constraints. These 9 solutions came close to feasibility which indicates that the optimization was capable of finding an adequate POF performance region. These results are shown in Table 5.13.

Table 5.13 - POF at generation 80

Worst case Phase Noise 10MHz (dBc/Hz)	Worst case Power (W)	# Failed Constraints	Worst case Phase Noise 10MHz (dBc/Hz)	Worst case Power (W)	# Failed Constraints
-130.79	2.93E-04	12	-130.30	2.46E-04	24
-130.78	2.68E-04	12	-130.15	2.18E-04	6
-130.61	2.63E-04	6	-129.84	2.11E-04	6
-130.51	2.59E-04	12	-129.74	2.02E-04	6
-130.46	2.57E-04	18	-129.59	1.91E-04	30
-130.45	2.54E-04	12	-129.13	1.79E-04	6
-130.44	2.51E-04	18	-128.77	1.64E-04	24
-130.31	2.46E-04	24	-126.99	1.60E-04	42

After the last 20 generations, the optimization found 12 feasible solutions near the region of performances at generation 80. It is evident that both worst case phase noise at 10 MHz and worst case power improved substantially when comparing with the original. The lowest and highest worst case power decreased by 0.1 mW and 0.46mW, respectively, while the lowest and highest worst case phase noise at 10 MHz decreased by 1.8 dBc/Hz and 0.32 dBc/Hz, respectively. These results are shown in Table 5.14.

Table 5.14 - POF at generation 100

Worst case Phase Noise 10MHz (dBc/Hz)	Worst case Power (W)
-130.88	2.45E-04
-130.51	2.32E-04
-130.50	2.30E-04
-130.42	2.26E-04
-130.16	2.18E-04
-129.73	2.13E-04
-129.72	2.12E-04
-129.64	2.04E-04
-129.54	1.88E-04
-129.53	1.88E-04
-128.99	1.82E-04
-128.99	1.82E-04

Once again, the speed-up obtained with the controlled PVT estimator is calculated by using the percentage of usage of each ANN throughout the optimization. These values are shown in Fig. 5.17.

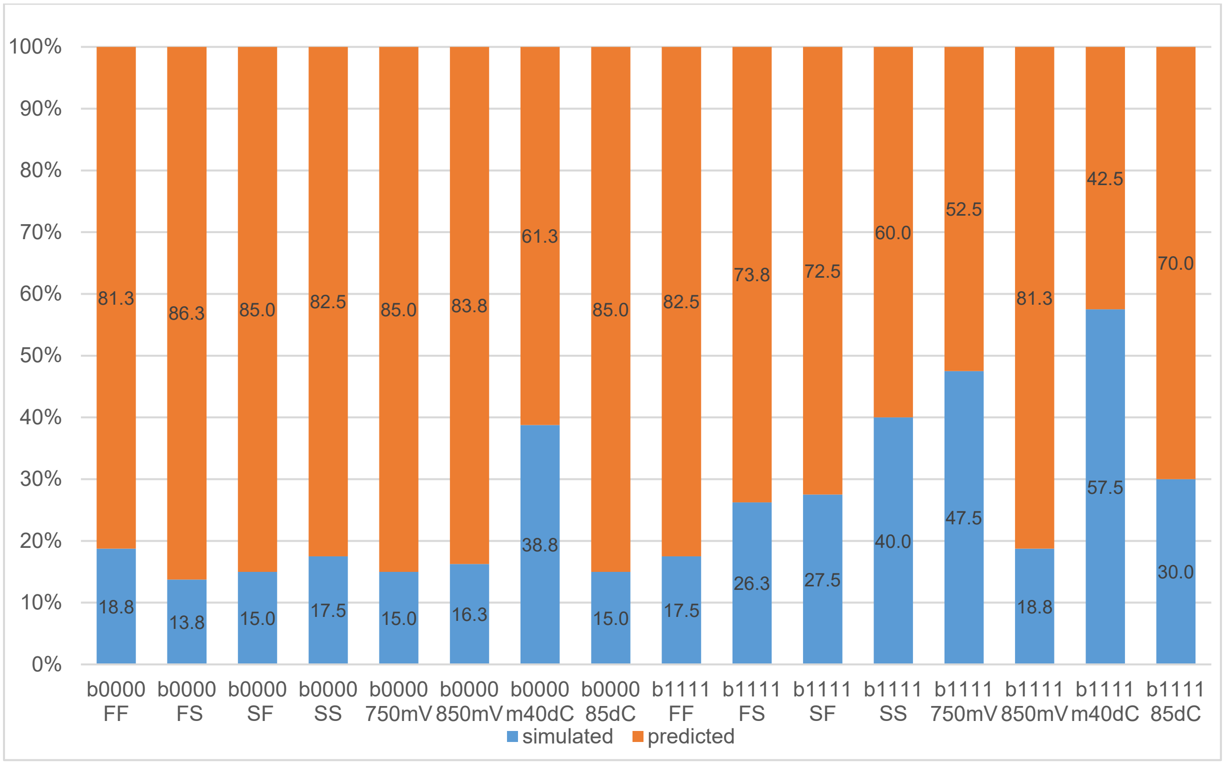


Figure 5.17 - Modes simulated vs predicted throughout optimization

When analyzing Fig. 5.17, it is possible to report that throughout the first 80 generations of the optimization 74.1% of the original PVT corners evaluations were instead predicted using ANNs. Considering the simulation of the TT conditions, using the controlled PVT estimator resulted in a speed-up factor of 2.92.

However, the total speed-up of the 100 generations is lower because the last 20 generations of full simulation must be accounted for, and the percentage of simulation added of these generations is bigger when compared to the previous optimizations (20% of total optimization). Considering this, the total speed-up factor obtained using the controlled PVT estimator is 2.11.

Conclusion

These results confirm that the controlled PVT estimator is capable of finding adequate circuit sizing performance regions despite its ANNs not being tuned for this optimization, while still being able to obtain a speed-up factor of 2.92. In the original optimization, the first 80 generations took, approximately, 509 hours to complete, while the optimization using the PVT estimator only took 174 hours to complete. The additional 20 generations using the original unmodified loop at the end of the optimization achieves overall better results than the original optimization, while still capable of achieving a speed-up factor of 2.11. With the PVT estimator, in total, the modified optimization took almost 14 days less than the original optimization. Once again, this reduction reveals the competitiveness in terms of computational time used to obtain feasible solutions.

6 Conclusions and Future Work

This Chapter presents the conclusions of the work done in this dissertation and outlines future possible developments which can possibly optimize the performance and abstraction of the PVT estimator.

6.1 Conclusions

In this work, it is presented an approach towards the acceleration of analog/RF IC optimization-based sizing loop with the help of a PVT corner performance estimator, using multiple ANNs, to complement the simulation process, therefore reducing the simulator workload.

For the development of the PVT estimator, an optimization-based sizing of a Class C/D VCO for 3.9-to-4.8 GHz was used as case study, gathering the necessary data to train the ANNs and to ascertain if the results of the estimations, before integration in the optimization loop, were adequate. All ANNs showed that the estimation error results were adequate so the integration process and discussing of final results were performed.

Three different circuit optimizations were used to test the PVT estimator. The first one was the same optimization based-sizing of a class C/D VCO for 3.9-to-4.8 GHz used for the development of the PVT estimator. The PVT estimator reduced 78.5% of the simulator workload, lowering the total optimization run time by 16 and a half days (original run took 25 and a half days to complete). The final solution results showed similar performances to the original optimization, and therefore, proving that the PVT estimator is capable of finding adequate PVT corner performances. The second optimization was an optimization-based sizing of the same circuit topology as the previous one, although, the range in which the VCO operates was changed to 2.3-to-2.5 GHz and the optimization constraints were tightened. The PVT estimator reduced 74.5% of the simulator workload, lowering the total optimization run time by 8 and a half days (original run took 14 and a half days to complete). Feasible solutions were found at the end of the optimization using the PVT estimator, proving its capability of being reused for optimizations with completely different targets of the same circuit topology its ANNs were trained to, therefore demonstrating its plug-and-play functionalities. The third and final experiment was an optimization-based sizing of a different VCO circuit topology, i.e., an ultralow power class B/C VCO. The same structure of the ANNs used in the two previous tests, was reused to train the ANNs in this optimization. The PVT estimator reduced 74.1% of the simulator workload, lowering the total optimization run time by 14 and a half days (original run took 26 and a half days to complete). Feasible solutions with better performances than the original optimization, were found at the end of the optimization using the PVT estimator, proving the capability of its ANNs reusage for a different VCO circuit topology, therefore demonstrating its plug-and-train functionalities.

6.2 Future Work

This work proved that optimization loop using the PVT estimator can achieve feasible solutions, while obtaining great speed-ups, however it is possible to further optimize the competitiveness in both optimization times, quality of estimation and the overall generalization of the PVT estimator when implemented in different circuit topologies.

While it was possible to prove that the modified loop can achieve feasible solutions, given the random nature of the optimization loop, each optimization can output different final solutions from another. Therefore, to check if the final solutions are indeed better or worse than the original optimization, several optimizations must be performed. As these optimizations would take a lot of time, several weeks or months should be spent on this process.

The datasets used in this work were obtained by executing optimizations, with the unmodified loop, with an execution time of almost one month. Using larger datasets with a small number of *null* and duplicated values would surely increase the estimation capability of the ANNs and given that smaller error results between estimation and real value would make the PVT estimator controller allow the ANNs to predict more often, the optimization would be faster, and the final solutions would be better.

An important aspect of the optimization with the PVT estimator is its great reduction of execution time. One hypothesis that it was not performed, due to time constraints, was making the execution time of the optimizations with the PVT estimator the same as the original ones, to verify if the final POF results would be better, given that in certain cases the number of generations would triple in number. A similar case happened in Section 5.2.3 where the POF results from the modified optimization were better than the original ones.

An augmentation of the capability of generalization of the PVT estimator could be done in two different ways: using datasets with data from different circuit topology optimizations and making the PVT estimator online with the optimization. The first one surely would need a more complex structure for the ANNs used in the PVT estimator, however there is a possibility that the ANNs would learn more information given different circuit topology data. The second one is an approach that uses the ANNs in such a way that a training phase would not be performed prior to the integration in the loop, therefore a dataset would not be needed. The ANNs would simply be integrated in the loop and for the first generations the loop would be working without them. At each generation, the ANNs would use the data from the input and output of the simulator for training purposes, therefore learning the circuit topology. In each generation, the ANNs would estimate some candidate sizing solutions and, using the real values from the simulator, the estimation error would be checked. If it was low enough, the ANNs would become online on the modified loop at that start of the next generation, beginning the speed-up process only then. With this the total speed-up obtained would decreased considerably but a time-consuming optimization, to gather the data for the dataset, would not be needed.

References

[1] N. Lourenço, R. Martins, and N. Horta, “Automatic Analog IC Sizing and Optimization Constrained with PVT Corners and Layout Effects”, Springer, 2017. Hardcover ISBN 978-3-319-42036-3.

[2] G. E. Gielen, and R. A. Rutenbar, “Computer-aided design of analog and mixed-signal integrated circuits,” in *Proceedings of the IEEE*, vol. 88, no. 12, pp. 1825-1852, Dec. 2000.

[3] G. G. E. Gielen and R. A. Rutenbar, "Computer-aided design of analog and mixed-signal integrated circuits," in Proceedings of the IEEE, vol. 88, no. 12, pp. 1825-1854, Dec. 2000.

[4] G. Gielen, “CAD tools for embedded analogue circuits in mixed-signal integrated systems on chip,” in *IEE Proceedings on Computers and Digital Techniques*, vol. 152, no. 3, pp.317–332, May 2005.

[5] Cadence. Accessed: Out. 12, 2021. [Online]. Available: http://www.cadence.com.

[6] MunEDA. Accessed: Out. 12, 2021. [Online]. Available: http://www.muneda.com.

[7] R. Martins, N. Lourenço, N. Horta, J. Yin, P. Mak and R. P. Martins, “Many-Objective Sizing Optimization of a Class-C/D VCO for Ultralow-Power IoT and Ultralow-Phase-Noise Cellular Applications,” in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 1, pp. 69-82, Jan. 2019.

[8] R. Martins *et al.*, “Design of a 4.2-to-5.1 GHz Ultralow-Power Complementary Class-B/C Hybrid-Mode VCO in 65-nm CMOS Fully Supported by EDA Tools,” in *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 11, pp. 3965-3977, Nov. 2020.

[9] R. Martins, N. Lourenço, S. Rodrigues, J. Guilherme and N. Horta, “AIDA: Automated analog IC design flow from circuit level to layout,” *2012 International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Seville, 2012, pp. 29-32.

[10] N. Lourenço, R. Martins, M. Barros, N. Horta (2013) Analog Circuit Design Based on Robust POFs Using an Enhanced MOEA with SVM Models. In: M. Fakhfakh, E. Tlelo-Cuautle, R. Castro-Lopez (eds) Analog/RF and Mixed-Signal Circuit Systematic Design. Lecture Notes in Electrical Engineering, vol 233. Springer, Berlin, Heidelberg.

[11] M. G. R. Degrauwe *et al.*, “IDAC: an interactive design tool for analog CMOS circuits,” in *IEEE Journal of Solid-State Circuits*, vol. 22, no. 6, pp. 1106-1116, Dec. 1987.

[12] F. El-Turky and E. E. Perry, “BLADES: an artificial intelligence approach to analog circuit design,” in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 6, pp. 680-692, June 1989.

[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*, vol. 9, no. 2, pp. 113-125, Feb. 1990.

[14] G. Jusuf, P. R. Gray and A. L. Sangiovanni-Vincentelli, "CADICS-cyclic analog-to-digital converter synthesis," *1990 IEEE International Conference on Computer-Aided Design. Digest of Technical Papers*, Santa Clara, CA, USA, 1990, pp. 286-289.

[15] G. G. E. Gielen, H. C. C. Walscharts and W. M. C. Sansen, "ISAAC: a symbolic simulator for analog integrated circuits," in *IEEE Journal of Solid-State Circuits*, vol. 24, no. 6, pp. 1587-1597, Dec. 1989.

[16] N. Lourenco and N. Horta, "GENOM-POF: Multi-Objective Evolutionary Synthesis of Analog ICs with Corners Validation," in *Genetic and Evolutionary Computation Conference*, Philadelphia, USA, Jul. 2012.

[17] R. Martins, N. Lourenço and N. Horta, "LAYGEN II: Automatic Analog ICs Layout Generator based on a Template Approach," in *Genetic and Evolutionary Computation Conference*, Philadelphia, USA, Jul. 2012.

[18] Hspice. Accessed: Out. 12, 2021. [Online]. Available: https://www.synopsys.com/implementation-and-signoff/ams-simulation/primesim-hspice.html.

[19] Calibre. Accessed: Out. 12, 2021. [Online]. Available: https://eda.sw.siemens.com/en-US/ic/calibre-design/.

[20] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation,* vol. 6, pp. 182-197, Apr. 2002.

[21] Spectre Simulation Platform. Accessed: Out. 12, 2021. [Online]. Available: https://www.cadence.com/ko_KR/home/tools/custom-ic-analog-rf-design/circuit-simulation/spectre-simulation-platform.html

[22] Eldo. Accessed: Out. 12, 2021. [Online]. Available: https://eda.sw.siemens.com/en-US/ic/eldo/

[23] K. P. Murphy, Machine learning: a probabilistic perspective (adaptive computation and machine learning series). MIT Press., 2012.

[24] M. Bayes and M. Price, "An Essay towards Solving a Problem in the Doc-trine of Chances. By the Late Rev. Mr. Bayes, F. R. S. Communicated by Mr. Price, in a Letter to John Canton", in *Philosophical Transactions* (1683-1775), vol. 53, pp. 370–418, 1763.

[25] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain", in *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors", in *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[27] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statis-tical Learning, The Elements of Statistical Learning Data Mining, Inference, and Prediction, Second Edition, 2009.

[28] Javatpoint. Accessed: Out. 12, 2021. [Online]. Available: https://www.javatpoint.com/regression-vs-classification-in-machine-learning.

[29] A. Suissa and et. al., "Empirical method based on neural networks for analog power modeling," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 5, pp. 839–844, April 2010.

[30] N. Kahraman and T. Yildirim, “Technology independent circuit sizing for fundamental analog circuits using artificial neural networks,” in Ph.D. Research in Microelectronics and Electronics (PRIME). IEEE, pp. 1–4, 2008.

[31] K. Zhu and et. al., “Genius route: A new analog routing paradigm using generative neural network guidance,” in *Proceedings of International Conference on Computer Aided Design (ICCAD)*, 2019.

[32] M. Andraud, H. Stratigopoulos, and E. Simeu, “One-Shot Non-Intrusive Calibration Against Process Variations for Analog/RF Circuits,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 63, no. 11, pp. 2022–2035, Sept. 2016.

[33] G. Wolfe and R. Vemuri, “Extraction and use of neural network models in automated synthesis of operational amplifiers,” in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 2, pp. 198-212, Feb. 2003.

[34] T. O. Çakıcı, G. İslamoğlu, Ş. N. Güzelhan, E. Afacan and G. Dündar, “Improving POF Quality in Multi Objective Optimization of Analog ICs via Deep Learning,” *2020 European Conference on Circuit Theory and Design (ECCTD)*, Sofia, Bulgaria, pp. 1-4, 2020.

[35] G. İslamoğlu, T. O. Çakici, E. Afacan and G. Dündar, “Artificial Neural Network Assisted Analog IC Sizing Tool,” *2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Lausanne, Switzerland, pp. 9-12, 2019.

[36] K. Hakhamaneshi, N. Werblun, P. Abbeel and V. Stojanović, “BagNet: Berkeley Analog Generator with Layout Optimizer Boosted with Deep Neural Networks,” *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Westminster, CO, USA, pp. 1-8, 2019.

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

[38] Hongzhou Liu, A. Singhee, R. A. Rutenbar and L. R. Carley, “Remembrance of circuits past: macromodeling by data mining in large analog design spaces,” *Proceedings 2002 Design Automation Conference (IEEE Cat. No.02CH37324)*, New Orleans, LA, USA, pp. 437-442, 2002.

[39] N. Lourenço *et al.*, "On the Exploration of Promising Analog IC Designs via Artificial Neural Networks," *2018 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Prague, pp. 133-136, 2018.

[40] N. Kahraman and T. Yildirim, "Technology independent circuit sizing for fundamental analog circuits using artificial neural networks," *2008 Ph.D. Research in Microelectronics and Electronics*, Istanbul, pp. 1-4, 2008.

[41] E. Dumesnil, F. Nabki and M. Boukadoum, "RF-LNA circuit synthesis using an array of artificial neural networks with constrained inputs," *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, Lisbon, pp. 573-576, 2015.

[42] N. Takai and M. Fukuda, "Prediction of element values of OPAmp for required specifications utilizing deep learning," *2017 International Symposium on Electronics and Smart Devices (ISESD)*, Yogyakarta, pp. 300-303, 2017.

[43] N. Lourenço *et al.*, "Using Polynomial Regression and Artificial Neural Networks for Reusable Analog IC Sizing," *2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Lausanne, Switzerland, pp. 13-16, 2019.

[44] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients", Neural Networks, vol. 21, no. 4, pp. 682–697, 2008.

[45] D. Silver and et. al., "Mastering the game of Go without human knowledge" in *Nature*, vol. 550, no. 7676, pp. 354–359, October 2017.

[46] Lil'Log. Accessed: Out. 12, 2021. [Online]. Available: https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html.

[47] Z. Zhao and L. Zhang, "Deep Reinforcement Learning for Analog Circuit Sizing," *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, Sevilla, pp. 1-5, 2020.

[48] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi and B. Nikolic, "AutoCkt: Deep Reinforcement Learning of Analog Circuit Designs," *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, pp. 490-495, 2020.

[49] F. Bre, J. Gimenez and V. Fachinotti, "Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks" in *Energy and Buildings*, vol. 158, November 2017.

[50] Aurlien Gron. 2017. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (1st. ed.). O'Reilly Media, Inc. ISBN 978-1491962299.

[51] B. Xu, N. Wang, T. Chen and M. Li, "Empirical Evaluation of Rectified Activations in Convolutional Network ", *Cornell University arXiv:1505.00853*, November 2015.

[52] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs) ", *Cornell University arXiv:1511.07289,* November 2015.

[53] "Understanding learning rates and how it improves performance in deep learning". Accessed: Out. 12, 2021. [Online]. Available: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10.

[54] "Early Stopping with PyTorch to Restrain your Model from Overfitting". Accessed: Out. 12, 2021. [Online]. Available: https://medium.com/analytics-vidhya/early-stopping-with-pytorch-to-restrain-your-model-from-overfitting-dce6de4081c5.

[55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting" *Journal of Machine Learning Research 15*, pp. 1929-1958, June 2014.

[56] Tensorflow. Accessed: Out. 12, 2021. [Online]. Available: www.tensorflow.org

[57] Keras. Accessed: Out. 12, 2021. [Online]. Available: https://github.com/fchollet/keras