TÉCNICO LISBOA

CAD: AIDA-TOP - Analog IC Topology Selection

João Maria Lobo Moutinho Soares de Melo

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Nuno Cavaco Gomes Horta

Prof. Ricardo Filipe Sereno Póvoa

Examination Committee

Chairperson: Prof. Teresa Maria Sá Ferreira Vazão Vasques

Supervisor: Prof. Nuno Cavaco Gomes Horta

Members of the Committee: Prof. Rui Santos Tavares

July 2020

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

Acknowledgements

It is impossible to thank all the important people that helped me during my master's degree in electrical engineering. Directly or indirectly, many people allowed the conclusion and enjoyment of this stage of my life and I will be forever thankful.

To my supervisor Ph. D. Nuno Horta for his guidance and help. To my co-supervisor Ricardo Póvoa who went above and beyond his duty in his support. My gratitude to ICG team, António Canelas for his diligent support, as well as to Ricardo Martins and Nuno Lourenço.

To my Técnico friends, to all my closest friends, and to my family, all of whom I shared many great moments with and that were always there for me. In particular I would like to thank those that had a special role throughout the completion of this dissertation, Daniel Farinha, João Ferreira, João Caeiro, Rita Mourão, José Mourão, Rita Lopes Mourão, my uncle Pedro Medeiros, my sister Luísa, and Monika Luśnia.

Finally, but not least to my parents, who always did everything in their power to provide me an extraordinary life.

This work was developed at Instituto de Telecomunicacoes and was funded by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/EEA/50008/2020.

Resumo

Esta tese de mestrado descreve o desenvolvimento de uma metodologia de selecção de topologias altamente customizável, independente da tecnologia, compatível com a ferramenta AIDA-C. A síntese de topologias é um passo essencial no decorrer do projecto de circuitos, maioritariamente dependente do conhecimento de desenhadores experientes. Criar uma ferramenta de EDA para realizar esta tarefa melhoraria substancialmente o *time-to-market*, visto que muitos outros elementos do projecto de circuitso já se encontram automatizados. O método escolhido recorre à informação de optimizações de dimensionamento anteriores, combinando-as com algoritmos de MCDM para obter a topologia mais apta, quer haja um circuito que já alcance os valores pedidos, ou caso não haja nenhum que o faça.

O método de selecção de topologias foi implementado no programa AIDA-TOP. Foi testado recorrendo a optimizações do AIDA-C de quatro topologias distintas, pertencentes à família OTA CMOS. A progressão do programa foi verificada usando variados exemplos de input. A ferramenta usou 40 optimizações de teste, cada uma contendo múltiplos circuitos simulados, atingindo 76,41% de taxa de acerto geral.

Palavras-chave

Projecto de circuitos analógicos; Síntese topológica; Automação de Projecto Electrónico; Selecção Topológica; Optimização Multiobjectivo; Análise de decisão multicritério.

Abstract

This Master thesis describes the development of a technology independent, highly customizable topology selection methodology compatible with AIDA-C. Topology synthesis is an essential step in circuit design, majorly dependent on the knowledge of experienced designers. Producing an EDA tool in this task of a circuit's design flow would drastically improve its time-to-market, since many other elements in this flow are already automated. The method chosen uses the information of past sizing optimizations, combining them with MCDM algorithms to get the most apt topology. It does so if there exists an optimized circuit able to reach the desired performances, or otherwise.

The topology selection method was implemented as the AIDA-TOP program. It was tested using AIDA-C optimizations of four distinct topologies, belonging to the CMOS OTA family. The program was verified to progress as expected with diverse input examples. The tool used 40 test optimizations with multiple simulated circuits each to arrive at an overall accuracy rate of 76,41%.

Keywords

Analog circuit design; Topology Synthesis; Electronic Design Automation; Topology Selection; Multi-objective optimization; Multi-criteria decision analysis.

Contents

List of Figures

List of Tables

Acronyms

ADC	Analog to Digital Converter
AMS	Analog or Mixed Signal
ASIC	Application Specific Integrated Chip
CAD	Computer-Aided Design
CMOS	Complementary Metal-Oxide-Semiconductor
EA	Evolutionary Algorithm
EDA	Electronic Design Automation
GA	Genetic Algorithm
GP	Genetic Programming
GUI	Graphical User Interface
HDL	Hardware Description Language
IC	Integrated Circuits
IOT	Internet of Things
KCL	Kirchhoff's' Current Laws
KVL	Kirchhoff's' Voltage Laws
MCDM	Multi Criteria Decision Makers
MINLP	Mixed Integer NonLinear Programing
MODA	Multi-Objective Decision Algorithms
MOEA	Multi-Objective Evolutionary Algorithm
MOO	Multi-Objective Optimization
MOSFET	Metal–Oxide–Semiconductor Field-Effect Transistor
NSGA-II	Nondominated Sorting Genetic Algorithm II
OPAMP	Operational Amplifier
OS	Output Swing Voltage
OTA	Operational Transconductance Amplifiers
POF	Pareto-Fronts

PSSR	Power Supply Rejection Ratio
SoC	System on Chip
SPICE	Simulation Program with Integrated Circuit Emphasis
TOPSIS	Technique For Order of Preference by Similarity to Ideal Solution
VOS	Offset Voltage

1 Introduction

This chapter is an overview on the of the work that has been done, laying out the reasons for its conception and the goals it sets to reach, while defining its place amongst the existing solutions. Then, it presents the document's structure, showing its logical flow, with the additional benefit of making it more searchable.

1.1 Motivation

Electronic circuits flood the contemporary world, taking part in our day-to-day, in a great variety of ways. In their infancy they existed scarcely as a tool to propel new means of communication, having grown with the appearance and popularization of radios, telephones, and televisions. With the invention of the transistor, and the ability to integrate it in large numbers into a semiconductor made integrated chip (IC), their adoption became mainstream. ICs have a cheap and reliable production method, making them ideal for mass production. This revolution made products such as computers, mobile phones and home appliances available for most consumers and infiltrated most areas of professional activity. To understand the magnitude of this expansion it can be observed, in Figure 1.1, that the semiconductor market worth grew from $20 billion to $480 billion [1], [2](approximately), from 1986 to 2018, sustaining the success of multiple companies amongst the most profitable ones in the world.

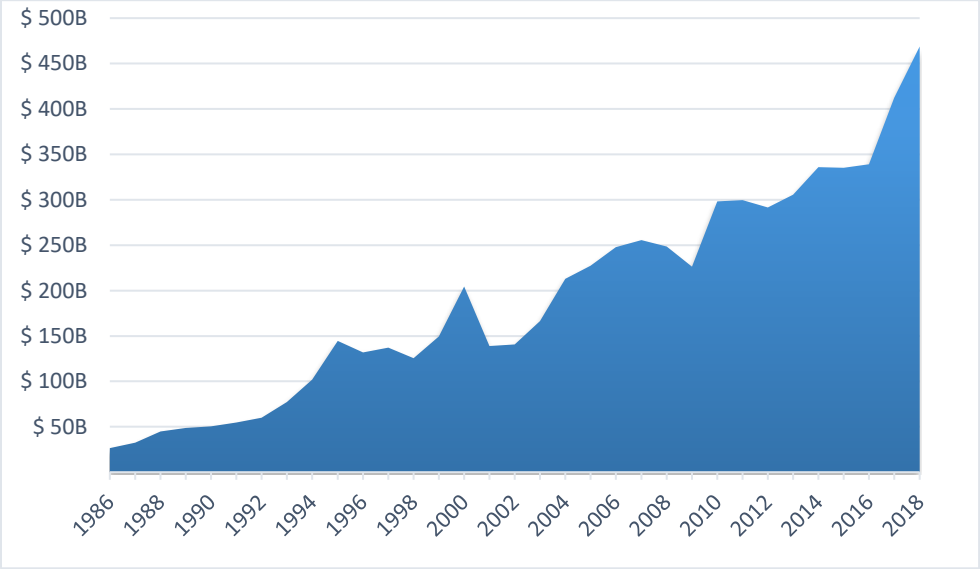

Figure 1.1 - Semiconductor Billings from 1986 until present.[1][2]

At first, the symbolic representation (schematic) and the physical implementation (geometric blueprint, also called layout) could be made manually. The number of devices that an integrated circuit contained grew exponentially, making it to dozens of billions nowadays [3], soon making their hand conception unviable, due to the number of devices and the precision required. Besides, the testing and correction of a circuit might require a long time and bear great costs.

Computer-aided design (CAD) surfaced to improve electronic design automation (EDA), facilitating the creation of circuits, in all its sectors from tools that ensure and implement logic, to ones that implement the physical circuit construction and verification of the layouts for fabrication. In the last 20 years processors, memories, digital and mixed-signal blocks etc. have been relegated to a single silicon substrate, the system on chip (SoC) that, on the digital front, often resorts to the reuse of blocks[4]. These blocks are application specific ICs (ASICs), cells that were standardized for a certain functionality. This uniformization made it possible to develop higher abstraction level, leading to the inception of hardware description languages (HDL), who are, through logic synthesis, converted in schematics. [5]

Such mechanisms, that greatly increased productivity, have not been employed in analog or mixed-signal circuits (AMS). This lack of libraries constituted by standard cells is one of the reasons why, despite the relatively small proportion of analog circuits in electronic devices, their design is more complicated than the digital counterpart. Complexity in analog circuits stems from the intricate relations between devices, and performance goals and specifications being continuous.[6] This trait means that they are more easily impacted by interference brought by the constant reduction in chip sizes. Moreover, digital circuits are, by nature, improved with the amount of computation power, directly linked to the number of transistors. In the AMS world, this is not applicable, removing relevance of standard cells and, consequently, from the description languages that can be originated.

The present difficulty in the development of AMS circuits, is a threat to product life cycles due to the increasing time-to-market-constraints. Furthermore, the task is resource intensive, especially in human form, since it relies on analog designers, who must be highly trained. Due to the continuous nature of the “real world”, AMS circuits emerge as a necessity to interface with digital systems. This connection not only makes them indispensable, but also intimately links the successes of IC and semiconductor market, with the ability to produce this category of circuits.

Figure 1.2 shows the largest grossing applications for analog ICs. A growing trend in the automotive application is visible, propelled by the investment in driving cooperation and autonomy, requiring a large number of sensors to make it possible. Communications have been this industry’s biggest sector and are predicted to continue leading it. These factors, in conjunction with up-and-coming technologies, like the Internet of Things (IoT), leave optimistic prospects for the analog market, as shown in Figure 1.3. This figure displays the compound annual growth rate (CAGR), a business measure of cumulative growth, in this case from 2018 to 2023. In comparison to other device types within the semiconductor industry, analog ones are forecasted to have second largest rate until 2023 (7.4%), preceded only by memory devices (7.8%) [7]. The future interest in this type of circuits, and existing limitations to their conception, justify the importance of improving AMS IC design.

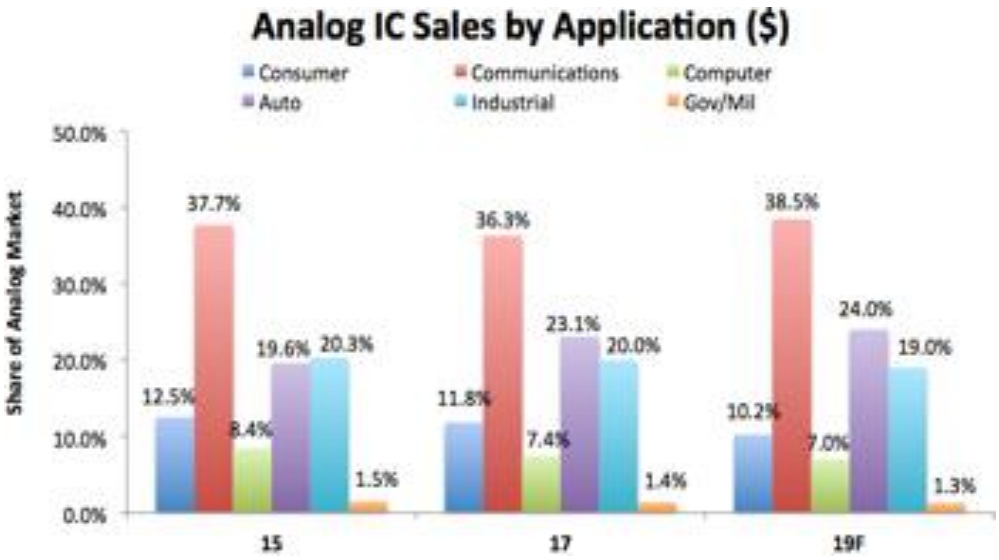

Figure 1.2 – Analog IC Sales by application in percentage of revenue, from 2015 to 2019 forecast. [7]

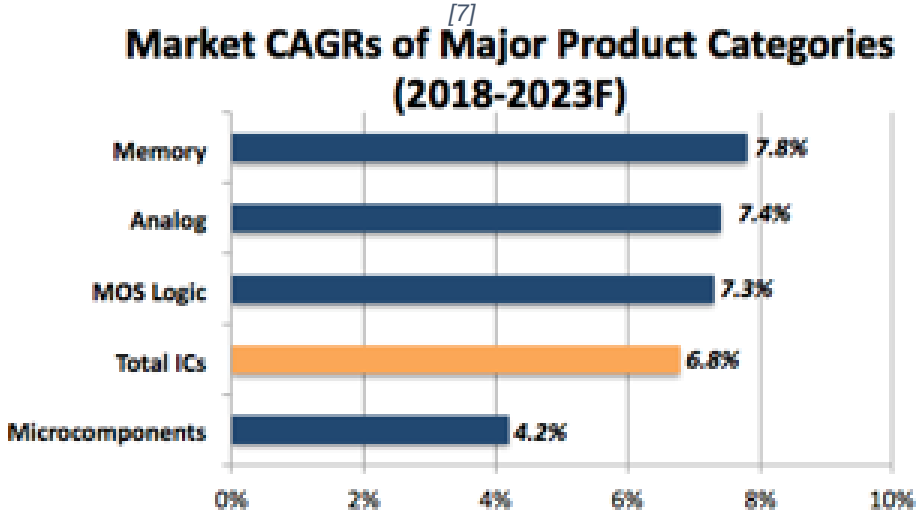

Figure 1.3 – Expected growth of product categories from 2018 to 2023. [7]

1.2 Analog IC Design flow

The need for bettering the design of AMS circuits has been established. This subsection will focus on describing the chain of processes to create a circuit. It will examine the status of AMS EDA and delineate the portion chosen the for intervention of this work.

The set of actions that start with the concept for a circuit and end with the delivery of a physical circuit is its design flow. Due to the differences explained in the previous section, even though the AMS and digital components will integrate the same chip, they will have parts of their flow that are distinct. Because our focus is on the analog branch, the generally accepted [8] model for it, that was proposed by Gielen and Rutenbar in [9], defines the following stages:

1. **Concept design –** The behavioral description of the chip, and the qualitative specifications to be reached are outlined.
2. **System design –** Definition of hardware and software elements and their languages (if applicable). The intended behavior of hardware components is described, as are the interfaces. Choice of technology, packaging, and test strategy for verification, are analyzed, on the implementation viewpoint.
3. **Architectural design –** Construction of the hardware architecture from the highest-level perspective. Segmenting the architecture into interconnected functional blocks, distinguishing analog and digital elements, while ascribing requirements to all components.
4. **Cell design –** After this step digital and analog design flows are not the same. Specification of analog blocks, detailing their topological design - the devices (transistors, resistors etc.) composing the blocks and the way they are connected - also attributing their parameters e.g. transistor physical dimensions, resistance of resistors. The latter activity is called circuit sizing.
5. **Cell layout –** Transformation of a sized electrical schematic into layout, geometrically parameterizing multiple layers, using the minimum area whilst avoiding parasitic effects that ruin performance.
6. **System Layout –** Create the higher-level layout that is reached by gathering all cells after they are defined, placing and routing them, putting together the IC. Establishes shielding mechanisms to guard from crosstalk and substrate noise coupling.
7. **Fabrication and Testing –** The molds that mirror the layout (masks) are generated and used to produce the ICs, the deliverable from the whole flow where the actual performance can be validated.

All the stages include simulation and validation to see if the requirements have been met. If in any of them are not met, it is possible to backtrack to a previous process, regressing in the flow, to correct mistakes or attempt alternative options.

This design flow for circuits represents a top-down approach, starting with an abstract perception of a full system, finishing with a concrete and detailed view. The higher levels (earlier stages) allow for a broader exploration of the search space, enabling a more varied gamut of circuits. For this conversion to happen, simplifications must be made about the components that integrate it, modeling their behavior. Models should sacrifice the minimum amount of accuracy whilst being as simple as possible, to allow rapid simulation. A model is deficient when: (1) it leads regularly to options that in lower layers will prove to be inaccurate, (2) is not simple enough. In both cases it will result in longer design times, by backtracking often in (1) and through excessive computational complexity in each run for (2).

To produce tools for higher levels, it is necessary to have sufficiently exact tools for the lower levels. Starting from the bottom, if fabrication and testing processes do not work properly, it is not feasible to extract, and model information to automate the chip's layout. The same is seen throughout the chain. So, if the layout

simulation is erroneous, the cell design will be as well. And if it is not automated, modifications in cell design, can require laborious layout redesign, due to the more concrete and detailed nature of this task.

The way EDA tools have developed supports this thesis. Initially came the automation of fabrication, which started in the 1960's [10]. The Cell Layout level has some solutions with commercial potential that start to surface (LAYGEN-II[11]), as does the sizing portion of Cell design (AIDA-CMK[12], Cadence's Virtuoso ADE GXLs[13], Synopsys' Titan ADX[14]) and some tools that already integrate both (AIDA[15]). However, when one examines the topological side of designing cells one confronts a task that still relies on graphical schematic entry tools, a manual chore relying on the expertise and time of highly trained analog designers. In view of this handicap, the current work proposes a methodology to improve the fourth stage of Cell Design. Namely, it is a proposal of a system to help in synthesizing a topology. Figure 1.4 provides a diagram view of the design flow pointing out the step where this project intervenes.

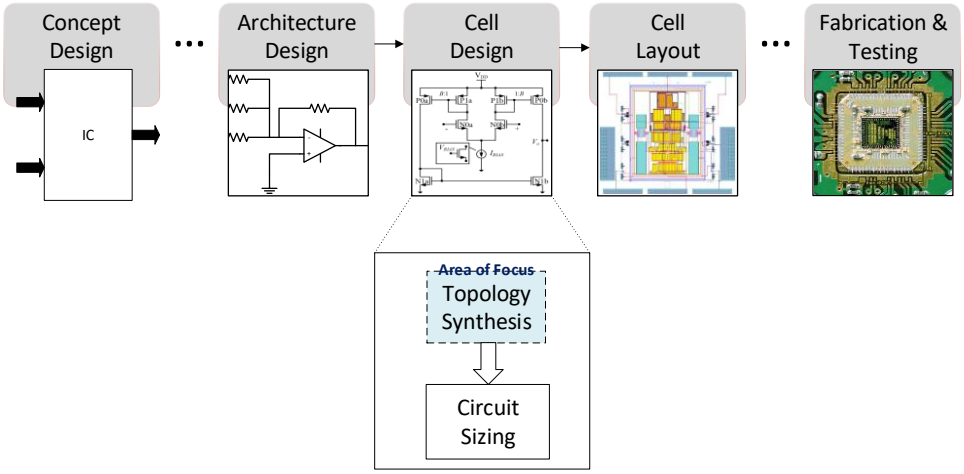

Figure 1.4 - Design Flow diagram exalting the area this work focuses on (Topology Synthesis).

Currently to get to a topology, an analog designer comes up with a schematic he finds to be likely to conform to the set objectives. This is done by using topologies that the designer has previously worked with, having acquired sensibility for their abilities, sometimes tweaking them to accommodate specifications that might be troublesome.

These topologies are often inserted, device by device, into a graphical schematic tool. Upon conclusion, the schematic can be input to sizing and layout tools, but at this point, the range of achievable performance specifications is much more narrow. For this reason, it is paramount, for successful circuit creation, that an adequate topology is chosen, as a failure to do so, limits the potential of a circuit drastically. If the topology is deemed unfeasible, redrawing is required, repeating the process until all goals are reached. The whole process is a time-consuming endeavor reliant on highly trained designers.

The integration of a tool that would automatically choose the schematic could, make this choice more efficient. Such a tool would automate the manual tasks and the expert knowledge involved, letting the expert designers focus on other tasks.

1.3 Topology level automation

This work intervenes in the first stage of cell design of electing a topology. It is a task of infinite possibilities, as it is possible to choose any number of components and combine them in different manners. It is further complicated by the strong effect and high variability that the devices parameters have.

To the action of experimenting combinations of elements roaming the design space coming up with atypical architectures the name of topology generation is given. This technique came up with the advent of genetic algorithms, allowing the creation of hundreds of thousands of unique structures[16]. Nevertheless, the portion of physically meaningful alternatives generated is minor, and the overhead of inspecting them reduces the attractiveness of this activity. Alternatively, with topology selection, tools often create a bank of solutions which designers know and understand, from which the tool chooses the one believed to have the largest potential to succeed. While selection began as the most popular approach, recently the trend has been to use the generation method.

Within these classifications, a wealth of ways to approach this problem have been attempted, some deficient due to excessive computational requirements, others due to the restrictive library, or due to the time necessary to compose a model. These libraries are largely sector dependent and reveal to be easily outdated by innovative technologies and designs. Recently a new paradigm was introduced where a balance between generation and selection is sought, originating new topologies exclusively if the library does not contain a viable solution. The strategy is explored by us is in topology selection.

To fill the breach in contemporary IC design, it is offered a methodology enabling the creation of a knowledge database (library), that can be customized with the most useful solutions for multiple sectors and uses. The hypothesis of using previous multi-objective optimizations is explored, to deduce which circuit can reach specifications currently required. To create that knowledge bank, Multi Criteria Decision Makers (MCDM) are adopted, summarizing the optimizations, into a smaller set of points that can describe them competently. Besides this functionality, an algorithm is described which will then parse through the stored information, selecting circuits that conform to the requirements of a newly asked circuit, suggesting the most flexible, performing topology. In case of not having a solution, the methodology uses the database to recommend the most promising design, so that further optimizations in sizing, or manual alterations to the components can be performed to reach the goals asked.

1.4 Objectives

The finality of the endeavor here presented, is divided into:

- Analyzing key academic works that represent major milestones regarding the development of topological synthesis aids.
- Creating a topology synthesis methodology, complementary to AIDA-C that:
    1. receives the desired performance for a number of goals and delivers circuits, from a library, capable of attaining them, in preferential order.
    2. In case of failure to encounter such circuits, delivers a list of available topologies based on their potential to be improved to fulfil these goals
- Choosing a collection of circuits and measurements suitable for performing optimizations on them to form the library.
- Developing a test set to verify the possibility of using optimizations with a higher number of objectives for lower number of objective requests to prove the tool's usefulness and correct functioning.
- Implementing the Topology into the AIDA-TOP tool.
- Examining the results of the application of the test set on the tool.

To define these objectives as successfully reached it is necessary to prove the methodology works. There are two manners to show the methodology functions correctly. The optimizations will be run on circuits that are known in-depth. This will allow for comparison between what would be expected before-hand, against the recommendations provided by the AIDA-TOP tool, which implements the methodology. Additionally, optimizations in which the objectives are subsets of the initially performed, will be executed, to evaluate to what extent the sizing with more goals can be indicative of a circuit's potential. This validation should be made in way to exclude statistical anomalies, providing a good confidence interval.

1.5 Document structure

This thesis will be divided into three main chapters. To start, a literature review, delving into the subject of choosing a topology. It will lay, chronologically, approaches previously attempted, selected to demonstrate different categories of strategies, and how they came to be. Still in Chapter 2, the categories stated are summed up, before they are used to do an extensive labelling of all previous works. On the 3rd Chapter the actual definition of our solution commences. In this section the details of the methodology are clearly defined, providing the reason that lead to the decisions made all through its creation. Chapter 4 logic and details of the preparations executed to check the precision of the methodology. Chapter 5 shows results achieved from using the tool, giving the metrics that can be used to discern whether this enterprise should be considered a success. In addition to these chapters it is written a final summary of everything accomplished, and ideas for the continuations to this work are given.

2 Previous Works

Choice of appropriate topology is a significant step for proper circuit CAD. This significance has made it a subject that has been analyzed throughout the years, searching for an adequate solution. Proposed strategies of approach diverge broadly in their results and methodologies. In this section, they are investigated, to understand which ones are most suitable, and thus could serve foundation to devise an improved solution. Furthermore, other knowledge requirements are presented, to fully comprehend the decisions made in constructing a novel algorithm, as well as its way of functioning.

Firstly, the main classification of solutions is explained, splitting the task into two main branches. Then a historic perspective is issued, where all projects that introduced a new type of paradigm are surveyed, exploring innovative concepts, which create new categories. Afterwards, all these categories and their use, are summarized to be applied in a table classifying a wide selection of works. This same table will include (when available), metrics that provide further details about each tool.

To understand the development of this framework, there are also some concepts, from related subjects that need to be explained. In this work multiple non-trivial concepts and algorithms such as Pareto-Fronts (POF), Multi-Objective Decision Algorithms (MODA), or nondominated sorting genetic algorithm II (NSGA-II) are used. They will be explained either where they first were applied during this review of previous works.

2.1 Topology Selection vs Generation

Pertaining to the objective of synthesizing a topology to reach a set of requirements, the mechanisms involved are typically distinguished into two disjunct categories: Topology Selection and Topology Generation. Similarly to the remaining classifications, it could also be presented within the context of the historical overview, nevertheless, its preponderance made it logical to emphasize, by describing it separately. While the overall perspective is that these are easily distinguishable, there is some misperception in the boundaries of each strategy. This confusion can be observed when sifting through past ventures in the subject of EDA and it stems from the usual meaning of these words, in their English language usage. In this subsection each of the strategies is demarcated clearly, making for a concise definition, that can be objectively applied.

As it has been mentioned before, when designing an analog circuit there is a never-ending number of topologies, that can be created by connecting multiple components in different manners. There are some architectures for each class that designers, depending on their specialization, know well. They are aware of their applications, limits, and general features. These types of topologies are finite yet numerous. The process of selecting the most appropriate topology amongst known ones, is named **Topology Selection**. Another way of approaching the problem is to construct one or more topologies, combining components or blocks of components. Here there is infinite room for expansion, by adding components or simply by joining them in previously unconsidered patterns. This is called **Topology Generation**.

The reason for misunderstandings is that some topology generation methods, originate a multitude of topologies, until all combinations of connections, up to a maximum number of components, are exhausted. These compose a library, from which the relevant ones are said to be *selected*[17],[18]. This wording, albeit correct, in everyday terminology uses the same word that was chosen to define the election of a schematic within those with a familiar structure. To ensure a clear distinction I would state again that Selection encompasses strictly habitual designs. If a topology, or wealth of topologies, stand outside of the bibliography of a class of circuits, then Topology Generation is being observed.

2.2 Chronological review

The idea of using computers to aid in the task cell design for analog circuits has been around since the end of the 1980's. To make a contribution in this field it is helpful to be aware of what was attempted before. This awareness can only be complete, considering what made the main actors choose a path, evaluating the merits and pitfalls of previous attempts. This way it is possible to avoid redundancy, take advantage of well performing concepts, and escape methods that proved less advantageous.

This topic, as with many others, is marked by divergent approaches. The emergence of these can usually be singled out in time, allowing us to present the algorithms, still used currently, in a chronological way. In this subsection, works are presented in this order, giving a causal perspective of their development, as well as insight of the historical trends. By clustering according to new concepts, it is possible to understand the main ideas of several works without delving into implementation details of all of them. The implementation can be improved, or features can be added, but the divisions presented here, make the most difference in the results of a project. Despite this, it is also useful to have a notion of the troubles and considerations encountered in implementing a tool, hence the detailed analysis of individual papers.

2.2.1 Manual CAD

Even though there exist some small ventures into computer synthesized topologies, the first notable work to use the aid of a computers in the problem of choosing a schematic was IDAC [19], in 1987. It came as the culmination of independent endeavors on different types of circuits. This interactive tool introduced the idea of Topology Selection. It started on the architecture level design offering a library of classes of circuits: operational transconductance amplifiers (OTAs), analog to digital converters (ADCs), quartz oscillators, amongst others.

Within each, multiple different schematics were available to be picked by the user. Consequently, the process of selection can be labeled as **manual**. To size the topologies, the tool solved equations, previously coded into IDAC. Built-in analyzers simulated the fully sized circuits and compared the results with input specifications. Thus, regarding the design flow IDAC attempted to cover the architectural and cell design stages. Within the latter stage the sizing and topology selection were made separately, which given our

focus on the topology, it can be categorized as **independent** topology design. When the specifications failed, the program executed new iteration with adjusted specifications.

This process was to be repeated, exhausting all possible schematics, finally announcing the ones that should be able to perform in the desired way, while warning of potential shortcomings they could have. Regarding those that failed to meet the requirements, suggestions of how they could potentially be improved, were provided. In the end, the circuits were presented in a form that could be used by a sister tool ILAC[20], to perform the layout design.

This work was pioneer, representing one of the first CAD programs to be developed. It was an ambitious project, intended to help in two full stages of design. In the sizing department it strived for complete automation. In both, architectural and topology design, it saved time by having an incorporated library with useful solutions, ready to be used. Furthermore, after analysis, it supplied multiple successful topologies, with information about potential faults, from which the user could choose.

However, although this first approach gave a first glimpse into the importance of having an auxiliary in finding a topology, it did so in a way that resorts to brute force, going through all chosen designs, indiscriminately. At that point the selection relied on the user to choose the relevant topologies and interpret the results of the sizing tool.

2.2.2 Hierarchical view

Two years later, Harjani et al. [21] developed a new system, OASYS, attempting to address IDAC's lack of flexibility due to the usage of circuit schematics that could not be altered, more difficulty to tune failures because of a less hierarchical structure, dependency on a separate numerical optimization phase for coarse design loop, and lastly its more closed goal, presenting only a series of programs for specific cases.

To resolve these issues, this solution ported an approach from the digital counterpart, with the caveat of the characteristic of reusage of portions cells, typical to analog design. The authors emphasized that a main focus of this new methodology is hierarchy.

Even though both IDAC and OASYS present hierarchies, they do it in a distinct manner. IDAC displayed it through coverage of several design flow stages, being overly ambitious in the breadth of stages it encompassed. This overreach is evidenced by the fact that future ventures commonly restricted themselves to one class of circuits. IDAC first focused on architectural design, which has classes of circuits, and cell-design (instance of a class), after the class is chosen. OASYS only considers the cell stage. However, it further subdivided a cell into sub-cell abstracts blocks (e.g. current mirror), sub-blocks.

Given that our work's interest is centered in cell-design, the approach is considered **hierarchical**, if it divides cells into more than one layer like in OASYS. On the contrary, IDAC's cell representation is **flat**, with each schematic being immediately composed of electrical devices. The use of hierarchy replicates the human designer's *modus operandi*, of partitioning the task into designing multiple smaller components. If the circuit is larger, extra levels may be added. The layers, their components, and respective models were created from designers' knowledge of often reused blocks and abstractions.

In this hierarchical structure (as seen in OASYS), not only is the circuit design broken into smaller pieces, but additionally, it permits the usage of the same blocks in different contexts, setting them up only once. Nevertheless, this representation removes the ability to employ design "tricks", that affect multiple blocks, which are only accessible when single devices are exposed and independently mutable, a capability this tool does not possess. Humans have the knowledge to implement such tricks, using a flattened view of the schematic, expanding the limits of performance.

In this same implementation iterations of two steps are needed to arrive at a complete circuit. The number of iterations depends on the layers of hierarchy that exist. For each layer, firstly a distinct combination of building blocks is chosen. These interconnections of blocks are called design styles. Then, from the known interactions between the blocks, and the final desired requirements, specifications are translated for each block. This couple of tasks is undergone until the design styles' blocks are elementary components. The first step is design style selection, and the second translation. As am example, to design an opamp, two iterations are performed. Initially, different connections of sub-blocks are considered, after choosing one, the specifications are translated so that the best combination of connections of transistors within a sub-block can be designated and finally sized. A visual representation of these concepts is shown in Figure 2.1. In the end, the framework strived to meet required specifications, allowing for optional future optimization.

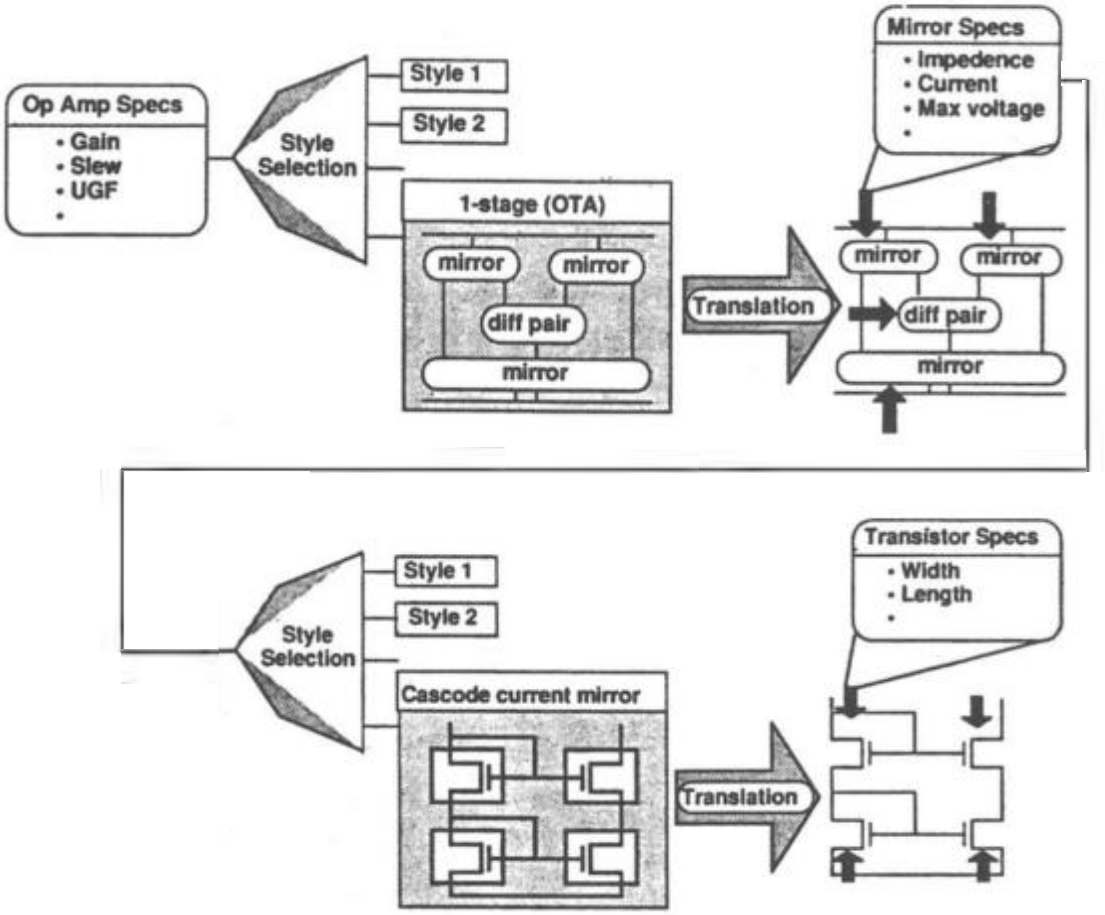

Figure 2.1 – Two-layer Translation and Style Selection steps for opamp design. [21]

With this strategy, the design task remained uniform, independently of the complexity of circuits. Furthermore, since block interconnections are limited, there was a fixed number of design styles. For the specification of each design style into device level, there existed also a fixed number of possibilities, albeit with variable component sizings. This represents a divide-and-conquer approach to the problem, making for balanced computation between tasks. Such structure also permits for scalability, as the process remains unchanged, only considering additional options, should it be decided to add sub-blocks, if each one is sufficiently defined.

2.2.3 Knowledge-Based implementation

While choosing design styles OASYS relied mostly on attempting multiple ones generating them in a predefined order, testing and storing the best. The thought behind this, was that a wealth of design styles can be coerced to perform at asked standards in different situations, even if sub optimally. The authors stated that it was a much more natural way to choose fully implemented circuits than using heuristics to

guess beforehand which one will be better. Despite this, some heuristics were also used to obtain an approximation of the feasibility of the critical parameters, permitting discrimination amongst design styles.

In the beginning, performance specifications were input by the user. After design style selection, the program now requires translation for defining specifications that the styles' constituent blocks must attend to. To execute the translation, a design style needs a design plan, consisting of useful circuit relationships. Since problems are often under-constrained, the storage of the plans, as basic analytical relationships, is often insufficient. The plan is a course of action to arrive at intended parameters. They consist of steps that apply heuristics that might need future verification e.g. estimation parasitic capacitance, gain partitioning in different stages. These also computes algebraic linear equations and sets of simultaneous equations.

A failure handling system is implemented, which is activated by the impossibility of achieving a certain specification with the default plan. Plan-fixers, chosen with IF-THEN rules, are either based on simple actions, or be more algorithmic. These fixes are required when some heuristics were used to get an approximation of a quantity, proved incorrect by a more precise computation executed in a lower level.

This tool further automates the process, by replacing extensive simulation and optimization of human selected circuits, with methods that reduce computation, while still providing the best or at least a good option. The authors instilled subject specific information, to be used for synthesis. This information must be manually modeled and introduced into the tool, making them **manually compounded knowledge-based** tools.

2.2.4 Joint design stages

Contrary to IDAC, in OASYS it is unfeasible to separate clearly what actions focus on topology creation from those that concern sizing. This option is antagonistic with the independent, regarding the separation of the cell design parts. When the design flow does not demarcate the topology design part, it can be defined as **joint** topology design, indicating the union of the topology design stage with other tasks.

In OASYS' particular case topology and sizing are mixed, but it is possible to find encounter others joining different stages, as architectural and/or layout. Whilst joint design presupposes that other tasks are considered, independent design, may or may not have additional ones. In IDAC it is seen a topology design independent stage, but sizing is executed afterwards.

2.2.5 Improvement of overall performance

At the same time as OASYS was published, OPASYN [22] was released. It set out to synthesize Complementary Metal-Oxide-Semiconductor (CMOS) Operational Amplifiers (OPAMPs) completely, covering cell and layout design. The component here scrutinized is the Selection Module, responsible for providing a promising topology, which will later be introduced into size and layout modules. Alternatively, the user could input a schematic of his choosing.

This work used flat view, citing an impossibility to propagate constraints within components, due to tight and intricate coupling between functional models. The 5 topologies, available in this work laid on the leaves of a decision tree shown in Figure 2.2. The choice was made by pruning it according to the demands in the following specifications: application area, open-loop gain, Power Supply Rejection Ratio (PSSR), and fully differential requirement.

Upon selection the topology would be forwarded to the sizing module where the parameters were tuned, maximizing a design cost function, calculated as a weighted sum of the performance in each parameter. This method of evaluating the success of a circuit departs from the previous works, that focused solely in attaining the constraints presented. It was then an **overall** performance objective, contrary to the previously seen focus on **constraints achievement**.

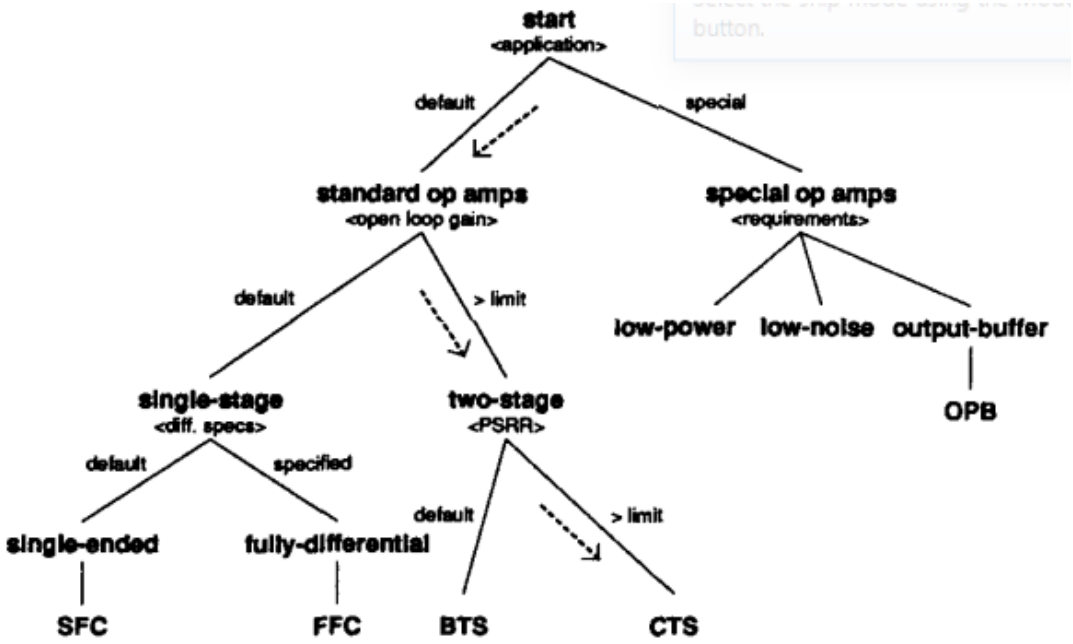

Figure 2.2 - OPASYN decision tree. [22]

At last, its layout was defined, and the final circuits presented to the user, that must make the final selection. Ultimately OPASYN's topology design appears to be overshadowed by the larger focus placed into the sizing and layout tools.

Even though both this work and OASYS were knowledge-based, it is possible to observe that they are distinct in the amount of knowledge they contain. This difference shows a trade-off within this type of solutions. OPASYN had only a handful of topologies available, limiting the choice when compared to OASYS. On the other hand, the implementation of OPASYN was simpler requiring only a few lines of code to choose between all options, whereas the other tool requires extensive codification per block. Generally,

as the options grow larger in this sort of solutions, so must the complexity of the code and the expertise and time required to program it.

In terms of hierarchy they present opposite mindsets, flat and hierarchical. The latter view provides the ability to generalize knowledge of a block into multiple usages, unbalancing the relationship between set up complexity and the number of synthesizable topologies. This advantage is gained by understanding all blocks and their relationships, which is demanding to do and even harder to implement.

2.2.6 Automating knowledge

Maulik et al. [23], realized that with a few adaptations, the problem of obtaining topologies and their device characteristics could be perceived as a Nonlinear Programming problem, thus enabling the usage of methodologies familiar this branch of mathematics. This was a solution joint with sizing, a point considered essential for this work.

Resorting to [24] cited in the paper, the definition of a nonlinear programing problem is formulated as:

$$\begin{aligned} &Minimize && f(x) && \\ &s.t. && g_i(x) \leq 0 && for\ i = 1, \dots, m \\ & && h_i(x) = 0 && for\ i = 1, \dots, l \\ & && x \in X && \end{aligned} \tag{2.1}$$

where $f(x)$ is the objective function to be minimized, varying the value of variables $x_1, \dots, x_n$ contained in the vector x belonging to subset of R^n, X, whilst obeying to inequality $g_i(\mathrm{x})$ and equality $h_i(\mathrm{x})$ constraints. To apply this formulation to the circuit design problem, it was defined that the variables had two meanings. Binary variables represented the inclusion or the exclusion of predefined parts of the circuit, e.g. common-source amplifier with the value of variable Y deciding whether it includes a cascode. Figure 2.3 shows the circuit with all options included, that is, with the cascode (dashed line) and the variable that controls it (Y) marked nearby. Additionally, to completely describe the circuit, the device parameters, whose representation is continuous, also needed to be included.

Usage of both, continuous and integer variables, leads to the description of this problem as Mixed Integer. Regarding the equations that constraint the system, they are not all linear, considering the circuits contain nonlinear components. Subsequently, the problem of solving for variables is classified as Mixed Integer NonLinear Programming reason for its name to be MINLP.

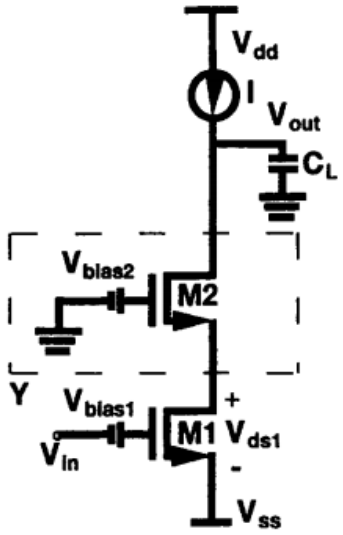

Figure 2.3 – Common-source amplifier with variable Y controlling the presence of a cascode.

In this work the constraints contained limitations of the technology, Kirchhoff's' Current and Voltage Laws (KCL and KVL) constraints, in conjunction with performance requirements. Variable conditions are trivial, performance conditions, however, required experts' time and knowledge to define. KVL and KCL were extracted resorting to an external algorithm, initiating the notion of **automatic compounded knowledge-based** tools, even if used only for part of the whole knowledge.

2.2.7 Fixed Single Objective

The objective function $f(x)$ in (2.1) was centered on minimizing the area of the circuit, the **fixed single** objective of this optimization solution. This goal, as the overall performance one, no longer arrests execution when constraint specifications are assured, continuing the search for designs that perform better in this sole objective.

The set of all solutions where the entirety of constraints is guaranteed is the feasible region (with feasible solutions in it), within which an optimum is sought. An algorithm was chosen that formerly demonstrated to be able to solve this faction of problems, was the elected method, adjusting the representation to be as convex as possible, improving the algorithm's performance. These steps are applied to 64 two-stage CMOS topologies, first setting up the circuits, then selecting of optimal one. MINLP is the only approach that explicitly states the problem as a nonlinear programing one, using a known methodology to discover optimal solutions.

2.2.8 Evolutionary Algorithms

A few decades after the birth of evolutionary algorithms (EA), analog designers began attempting to utilize its principles to automate circuit synthesis. Inspired by the successes of works which applied this subclass of biological-based algorithms in other steps of the analog synthesis flow, Lohn and Colombano [25] opted to include the task of defining a topology, representing the first impactful results in such effort.

Evolutionary computation bases itself on the Darwinian concept of survival of the fittest. It operates on collections of individuals, called populations, and much like nature, retains, on every generation i.e. iteration, only those who behave more fittingly to the surrounding conditions. As stated Ashlock's book [26] a algorithm is encompassed in evolutionary computation if it contains the loop in Table 2.1.

Table 2.1 – Evolutionary Algorithm Loop.

Generate population of structures
Repeat
    Test the structures for quality
    Select the structures to reproduce
    Replace old structures with new ones
Until satisfied

To implement such method, it is required that populations and their individuals are defined. Their structures must be detailed, the criteria by which the individuals are deemed fit set, and the necessary mechanisms to vary the population, creating different individuals that might be more adequate, employed.

In this version, the individuals that integrated the population were the circuits. The way each generation is ranked by fitness reflected the objectives of the algorithm, considering fitter the ones that performed better at them.

Also replicating biological evolution, the main mechanisms for diversification were crossover, and mutation. Crossover refers to the mixing of traits (genes) in surviving elements, an operation prone to generate healthy offspring, through combination of different characteristics. Mutation slightly alters an individual, potentially inserting an unexplored concept, aiding in exploring all possibilities. No justification was provided regarding the population size and number of generations set.

A circuit constructing bot placed components individually, in a manner that generated valid circuit graphs. The circuits had a cap of 150 devices, which could be transistors, resistors, inductors, and capacitors. The constructing thread decided which component was placed, how it was placed, and its parameters, encoding it into a bytecode. Since the device parameters were set simultaneous to the creation of the schematic, there was no division between the sizing and topology tasks. The bot moved along through a main branch,

extending the circuit one component at a time, creating occasional secondary branches. Consequently, a circuit was a list of bytecodes, defining the order of placement of all sized components.

This non-hierarchical view, centered on components, gave the ability to construct a wide range of topologies, only restricted by number of elements and validity insurance through limiting the main branch constructing thread to one node. This limitation had the benefit of keeping creation complexity at $O(n)$, by discarding the need for verification of circuit correctness.

To commence this algorithm, the initial population had to be created, having a user defined number of individuals up to 18 000. The circuit-constructing bot then generated all these circuits, providing a starting point for the algorithm. They were evaluated, computing the individual fitness, which was the aptness to reach the objectives.

According to [27], cited in the paper, the evolutionary algorithm in an iteration firstly undergoes reproduction, with a mating pool, having the same number of individuals as the current population, to be occupied with elements of the existing one. In this phase, copies of elements from the previous generation are chosen randomly, with the probability of an individual being replicated proportional to its fitness. These draws are performed independently until the pool is complete. This way, fitness imitates the natural ability of a creature to reproduce.

After these steps, the mating pool only contains clones of previous elements, not introducing diversity. Adaptable species often display genetic material from two parents, thus varying the gene pool. This concept is emulated by the crossover operator, combining two arbitrary elements in the new population through swapping portions of their genetic information, originating offspring. The rate of elements subject to this rearrangement is the crossover probability, or crossover rate[25].

Finally, the mutation process occurred, where a small percentage of values were changed by chance. The frequency of these alterations is the mutation rate. Both these methods work towards the exploration of the search space for more prolific solutions. After reproduction, crossover, and mutation a new generation is created. The crossover rate is typically much higher than the mutation probability, in tune with what happens in nature. This loop will be repeated until a set number of generations is reached. The listed steps, either the initial creation of the population, as well as subsequent generations, have the potential to originate unseen circuits, revealing the first topology generation tool.

2.2.9 Mutable Objective

In the MINLP (section 2.2.6) solution the concept of having a single metric, as objective to improve after all constraints had been met, was presented. However, in AMS design this objective is not universal. Even within the same class of circuits, an application, might require the largest gain, or lowest power consumption. As a reflection of this necessity, the latest work introduced (Lohn and Colombano [25]) doesn't set a particular goal for all classes of circuits. It gives the user the ability to select the specification considered a priority thus having a **mutable single** objective.

2.2.10 Circuit evaluation

Until this point, the way to evaluate the performance of a circuit, within its design, had been through numerical analysis of the circuits. The analysis is performed on circuits' models, requiring previous set-up of equations that describe it, either manually or with algorithmic aid. Different circuit categories demand distinct analyses to evaluate their behavior. The models contain approximations that can lead to relevant alterations in actual circuit behavior. This technique is **equation-based** evaluation.

On the other hand the work of Lohn and Columbano [25], uses the renowned Simulation Program with Integrated Circuit Emphasis (SPICE) tool. Released for public domain by Berkley University in 1975, this simulation program has been a staple from its creation to modern days a staple in electrical simulation. The program takes a standard circuit representation (netlist) and emulates precisely its physical behavior. The bytecode list format was translated into the netlist format for each of the individuals in a generation and ran by the simulator. It is possible to categorize the evaluation of circuits produced as **simulation-based**

Before this usage in the evaluation step, this simulator was usually run to verify a final design. The reason for only using in that stage is the nonlinear growth of computation requirements expended by SPICE with the growth of circuit nodes, becoming very demanding with this node increase. This characteristic is a consequence of the representation of circuits as nonlinear differential equations, whose solution is difficult but leads to an exact model. It is also a method that can be applied to generality of schematics, not necessitating further set-up beyond the translation to a netlist. Over the years other similar tools have emerged, particularly SPECTRE and Eldo®, having similar method of action.

In choosing a simulation-based method, accuracy and the ability to effortlessly use a wealth of schematics are chosen over computation speed. That is what happens in [25] with the added benefit of a parallel implementation, simultaneously simulating several circuits, reducing the population's evaluation time.

2.2.11 Classes of EAs

It is clear from the description of [25] that the work elaborates an EA. This class of solutions can be further subdivided into Genetic Algorithms (GA) or Genetic Programming (GP). The defining characteristics to place an approach into either category is based on the way of representation of the individuals. Koza [28], introduces GP as an extension of GA, looking to repair the limitation imposed through the exclusive usage of fixed-length character strings, expanding to more complex, adaptable hierarchic structures capable of encompassing computer functions. These should portray the hierarchy and dynamic size of a computer program, that can assume a tree form.

The project [25] assumed some qualities of a GA, as it had the typical structure of an evolutionary algorithm, and used strictly crossover and mutation. It also featured some relating to GP with its dynamic size, and the inclusion of function-like instructions for circuit placement. However, following the definitions provided, it failed the GA trait of fixed sized, and the GP's computer program hierarchical representation, leading the authors to present it as a GA with some GP elements. This new evolutionary methodology, through genetic programming or algorithm was groundbreaking and originated a great number of tools.

2.2.12 Genetic Programming

In the previous section the concept of genetic programming was introduced. In this section its first clear-cut, impactful implementation in topology generation is analyzed. The motivation behind the Sripramong et al. [17] was based upon the recognition of potential of evolutionary computation for analog circuit synthesis. It pointed out shortcomings displayed by the previously overviewed GA, which it vowed to improve. In particular it mentioned voluminous computation, from iterating 18,000 generations, the limitation of ways of connecting transistors, besides the lack of redundancy and unconventionality verifiers, creating repeated as well as unorthodox connections.

The schematics were represented in tree-form, conforming to the standard GP representation. The traversal of said tree instructed the evolution of a single wire from input to output into a complex network of device connections (Figure 2.4). For that, it had 5 different connection modifying actions (evolving functions) that a wire could be turned into, one node referencing function, 7 two-terminal components (resistor, capacitor, independent sources etc. as well as floating connection), and transistors (N or P-type) which have 3 terminals. A node could, for example, turn a wire into two parallel or series ones, and its leaves define which components were present in each of them. The parent defines how the leaves were connected, with exception of 3 terminal components that required a third connection and an evolving function that enabled the connection with a non-adjacent node, named a cross-linked connection.

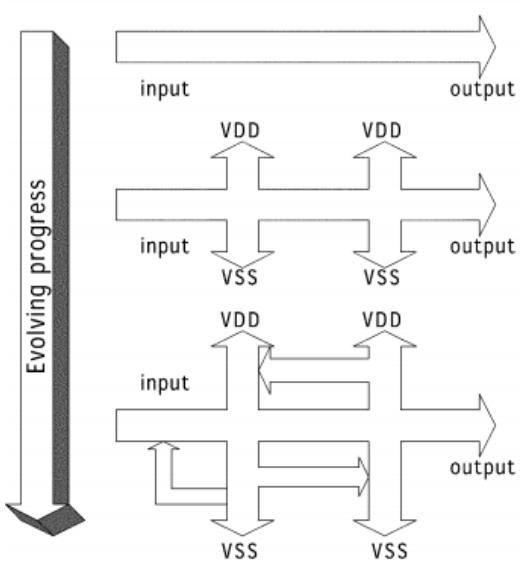

Figure 2.4 – Effect of evolving functions.

For every tree, current flow lists were extracted from their original representation, according to the dc current flowing through them. These lists were analyzed, to correct circuits with isolated or useless parts, change component connection, and obtain transistors' operating region. This methodology was based on Kirchhoff's laws, to process generated circuits. This verification ensured exclusively apt circuits were sent for PSPICE3 simulation, a resource demanding step, necessary to have precise data for the evaluation portion (simulation-based).

To initiate the evolutionary process the user had to supply an embryonic circuit. This circuit was converted to the program's preferred representation and cloned until it the population was complete. Afterwards, this uniform generation would undergo mutation, and current-flow analysis. The fitness function was composed as the normalized sum of 7 usual circuit goals, hence representing an overall performance goal, summing other 3 economic related targets after the main goals were reached.

The methods for developing new generations remain reproduction, crossover and mutation. But unlike the previous method of propagating the population, in this work, tournament selection is implemented, putting a few randomly selected individuals at odds, keeping the highest fitness one as a parent for crossover. Mutation is equally different, not changing just single bits, but resorting to the user-defined library to mutate random parts of a circuit and their parameters (mixed phases). This utilization of building blocks makes for a hierarchical representation of circuits. Regarding population size they contain 300-1000 elements, and the program halts when constraints are met, as opposed to pre-defining a number of generations

2.2.13 Multi-objective Optimization (MOO)

A multi-objective optimization problem is one that, as the name indicates, contains more than one objective function. A MOO problem is defined as [29]:

$$\begin{aligned} &Minimize && f_i(x) && for\ i = 1, \dots, M \\ &s.t. && g_i(x) \leq 0 && for\ i = 1, \dots, m \\ & && h_i(x) = 0 && for\ i = 1, \dots, l \\ & && x \in X \end{aligned} \tag{2.2}$$

Which is similar to (2.1), but instead of having one objective f function, there are M objective functions.. The concept of feasible region is also applicable in here. The challenge of finding an optimal analog circuit, can be seen as such, since designers usually have multiple criteria they wish to optimize e.g. minimize area and maximize gain. The strategies formerly shown to address this particularity, convert its functions into a single one. However, the principles of confronting a single function problem are dissimilar to facing additional objective functions.

One concept unique to MOO is Pareto-Optimal solutions. This is a MCDM, useful when there is more than one objective function, to determine optimum solutions.

Take a problem with $M = 2$, so it has 2 objectives. Choosing any two solutions from the feasible region, for some it can be noted that one is superior (dominating) to the other in all objectives (dominated). On the other hand, other pairs might have it so that one solution is better in the first objective but worse in the second. The same can happen but being superior in the second objective and inferior in the first. They are said to be non-dominated solutions, where none excels the other, if both goals are equally valued. The curve obtained from line-connecting these solutions, is the pareto-optimal front (POF) and the group of all solutions the Pareto Set. This formulation can be extended for a larger number of objectives. In Figure 2.5 there is an example of a pareto front and dominated points that are excluded from it. In this case the axes evolve in decreasing preference, so the preferred points are the minimums.

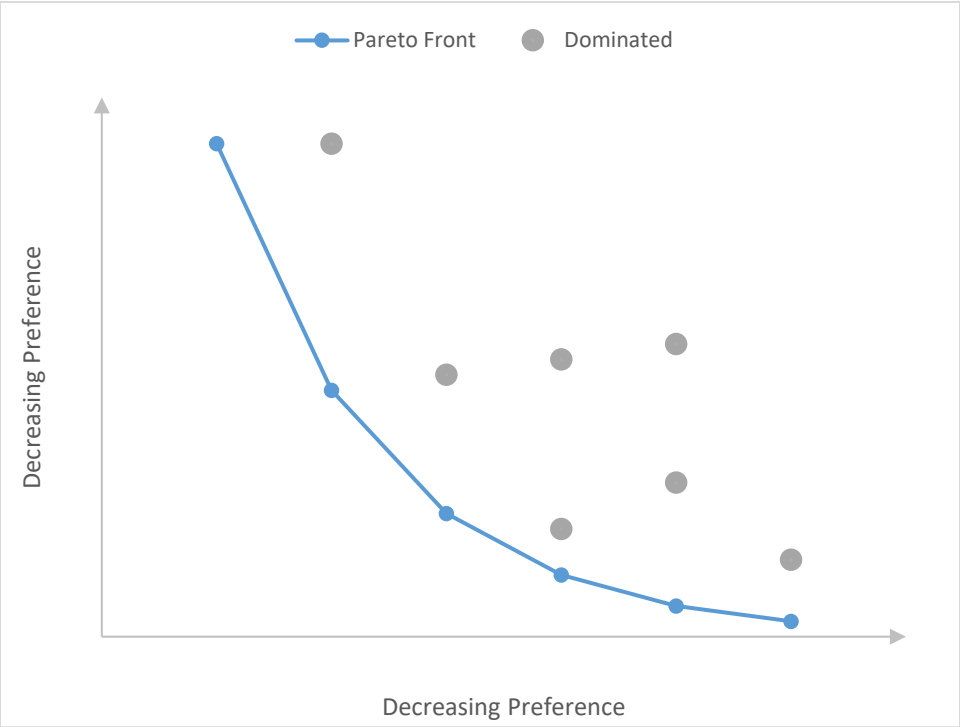

Figure 2.5 - Pareto Front example.

A multi-objective evolutionary algorithm (MOEA) joins components from the two subsets of approaches the Multi-Objective perspective with an Evolutionary basis for iterating to through the search space. MOJITO [16], published in 2011 does exactly this. A mixed phase, hierarchical solution, with emphasis in exclusively using trustworthy topologies connected in familiar ways, an area where the authors reckoned that previous tools faltered, leading to manufacture-fallible circuits.

2.2.14 NSGA-II

The MOEA applied in MOJITO [16] (for a number of objectives not much larger than 2) is the NSGA-II algorithm. The second version of the NSGA introduced in 2002 [30] became a staple for MOEA, for its application of elitism, diversity assurance and computational efficiency. Elitism is the act of storing the best elements (elite) from previous generations in subsequent ones, bringing advantages in terms of efficiency in addition to keeping the best solutions from worsening with generations[30]. At the same time, diversity is relevant by increasing the amount of pareto optimal-solutions[25]. The fitness metric here revolves on individuals not being dominated (nondominated) by others, to progress towards the Pareto-optimal region.

The algorithm has two important subroutines: 1) fast nondominated sort 2) crowding distance assignment. The first segments a population P, into fronts F_i where $i - 1$ denotes the number of elements that dominate all individuals contained. Crowding distance assignment routine obtains the average distance of an element to its neighbors, the smaller this distance, the more crowded and less diverse the population is. The main

routine, in Figure 2.6, initiates a random parent population P_0 with N individuals, on which tournament selection, crossover and mutation are applied, getting Q_0 an equally large offspring population. After, the main loop starts, both are joined into R_0 (2N elements) that will be sorted based on nondomination, obtaining fronts F_i. The elite fronts that cumulatively contain less than N elements will be indiscriminately added to the next parent set P_1. The front that would cause P_1 to exceed N elements will be sorted by decreasing crowding order $\prec_n$, to occupy the remaining spots (until N) in P_1 with the least crowded options. Q_1 is fabricated with the same evolutionary operations to obtain offspring as the ones used before. This loop is repeated for the desired number of generations.

$R_t = P_t \cup Q_t$	combine parent and offspring population
$\mathcal{F} = \texttt{fast-non-dominated-sort}(R_t)$	$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots)$, all nondominated fronts of R_t
$P_{t+1} = \emptyset$ and $i = 1$	
until $\lvert P_{t+1}\rvert + \lvert\mathcal{F}_i\rvert \leq N$	until the parent population is filled
$\texttt{crowding-distance-assignment}(\mathcal{F}_i)$	calculate crowding-distance in $\mathcal{F}_i$
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i$	include ith nondominated front in the parent pop
$i = i + 1$	check the next front for inclusion
$\text{Sort}(\mathcal{F}_i, \prec_n)$	sort in descending order using $\prec_n$
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - \lvert P_{t+1}\rvert)]$	choose the first $(N - \lvert P_{t+1}\rvert)$ elements of $\mathcal{F}_i$
$Q_{t+1} = \texttt{make-new-pop}(P_{t+1})$	use selection, crossover and mutation to create a new population Q_{t+1}
$t = t + 1$	increment the generation counter

Figure 2.6 – Pseudocode from [31] with the NSGA-II main loop.

MOJITO implements this algorithm, with some modifications to avoid premature convergence, in a GP based program. Namely the version used divides the population into age groups. In this manner, random individuals can be added to the population without being unfairly compared to circuits that already underwent improvements. This further diversifies the population, originating a broader spectrum of solutions.

2.2.15 Causal synthesis

Artificial cognitive systems emerged as a consequence of the interest in artificial intelligence looking to incorporate cognitive neuroscience and developmental psychology. The idea is to replicate the abilities of perceiving the environment, recognize the necessity to operate, gather experience from past events and mutate as required by the surroundings, into an artificial system. This sort of technique stems from a recognition of the superiority of humans in the performance of a wealth of tasks. To import what is described as unified theory of cognition, composed of skills such as attention, memory, problem solving, decision making, and learning, to a system is to create a Cognitive Architecture. [32]

Observing the problem of circuit synthesis, it fits the type of challenges that artificial cognitive system is sets out to solve. First and foremost, to this day the design of circuits is still reliant on expert-knowledge,

especially when it comes to the choice of schematic. Secondly, it is evident from the preponderance of hierarchy in previous works, that pattern detection is an important component. In addition to this, the ability to learn in these architectures could replace the manual or algorithmic methods of knowledge formulation once used. Li et al, presented in 2018 InnovA [33], a cognitive architecture applied to analog circuit design.

This cognitive architecture tries to exert creative problem solving, topology creation and modification, design knowledge identification and reuse. It appropriates ideas from the fields that have been referred as inspiration to these architectures, to bring functionalities like concept formation, comparison, and combination. It is a tool that generates innovative topologies, unbeknownst so far, through combination of concepts and features. This work, ultimately, tries to decompose circuits, understand the cause for the existence of its parts, and integrate these skills with the ability of learning to learn. The use of cause and effect for synthesis and learning introduces **causal** synthesis.

2.2.16 Conceptual View

Riddled with ideas from psychology and neuroscience, the InnovA architecture is intricate, having different modules coupled with each other. One of these, uses signal paths to autonomously detect building blocks, storing frequent connections between them, or small differences deeming them all as concepts. The uniqueness of these is ensured prior to addition. They are divided in clusters, based on topological similarity, each addressing a set of performance specifications (niche). The kernel gathers the common traits to the elements in the cluster, also containing what they add to the performance, thus supplying the reason for their presence (causality). This is the module of semantic memory, as the information stored is relevant within the same context (specific conditions). The union of all semantic memories is the long-term memory, which will originate short-term memory when using only a portion of it based on the task at hand and episodic memory, listing solutions for specific problems. The introduction of concepts that include the usage of traditional building blocks but not exclusively, also having small modifications that influence the circuit originate **conceptual view**, breaking from the preexisting dichotomy. This implementation simultaneously allows the advantage from both previous perspectives, reusing known blocks, but enabling the usage of “tricks”, all mentioned in section 2.2.2

2.2.17 Hybrid Generation and Selection

The architecture has a subjective and an objective reasoning and learning part. The reasoning occurs any time that the tool is presented with a problem. It runs through a series of mechanisms influenced by both parts to arrive at a solution. Long-term memory (and consequently the other types) is not static, it can be changed without the intervention of the user. This is the learning part of the architecture, having the ability to include circuits that the users supply. Furthermore, it can re-assess knowledge when presented with new challenges, by adding solutions or changing previous conceptions that had been created. Even though the purpose and way of operating of both are detailed, only the objective type and its algorithms are already implemented.

In the objective reasoning the following procedure is encountered: the short-term memory supplies a population of solutions with building-blocks used and their causality, creating a POF for the requirements needed. If one is found, the process is finished. Otherwise, the causality information is used to address requirements that are not fulfilled, identifying the features in a solution that might be less advantageous (small causality) and searching the long-term memory to get alternatives. This is called solution combination, which is done firstly by searching within the same cluster, then resorting to other clusters and finally through variation or analogy of the building blocks. When carrying out these operations, a projected outcome is guessed, based on the reasoning attached to each concept. The topology is simulated, obtaining a more precise evaluation of the performance.

After this part, the learning portion starts. Still on the objective perspective, if the results are largely dissonant with the expectations, it means that the knowledge needs to be refined. The ability to predict the performance of a solution is tightly coupled with the capacity to choose the right topology or creating new ones, hence requiring a modification in the assumptions that led to an incorrect prediction. Otherwise, if the results were coherent with the forecast, whether if it originated with the merging of elements between clusters, or from variants of existing concepts, that information is inserted into the long-term memory. Thus, the knowledge that the system contains is constantly growing, creating alternatives and improving the concepts' precision.

On the subjective section, the tool will be able to automatically detect the circumstances under which the causalities apply. The order of preference for similar features is also going to be decided here. Finally, emotions will dictate the degree of satisfaction with the matching between predictions and outcomes, varying with the quantity and nature of the existing errors. Additionally, the subjective part will also oversee instantiation, for each problem, of context-related memory, that is made of a subset of the subjective elements mentioned previously that are relevant to the issue at hand.

This recent tool resorts firstly to selection of topologies, generating multiple circuits only upon failure to encounter a job fulfilling circuit. This usage of both, similar to what is explained in the previous section, balances two opposite ideas, attempting to use the advantages of each. For its use of both it can be categorized as a **hybrid** solution regarding the discovery process.

2.3 Categorization of solutions

In this subsection an extensive inventory of works is presented, classifying them. The categories presented throughout the historical perspective are used for this activity, dividing them with respect to the views by which a circuit can be analyzed. Additionally, two quantitative metrics are included, to give a perception of the variety of topologies and real-world usage of the tools.

The categories, and their respective views are:

1. Discovery: Topology Selection, Generation or Hybrid.
2. Main Algorithm: Manual, Manual/Automatic Knowledge-Based, GA, GP, NSGA-II, Causal
3. Circuit Structure: Flat, Hierarchical, Conceptual View.
4. Circuit Evaluation: Simulation, Equation
5. Objectives: Constraint attainment, Fixed/Mutable Single Objective, Overall Performance, Multi-Objective.
6. Design Stages: The design stages also present in a work and if they are Independent or Joint with Topology Design.

The extensive listing of works is done in the form of a table that incorporates two metrics:

1. Number of topologies: Indicating the schematics from which selection must be made, or how many can be generated
2. Setup and Run Times: The temporal measure of the time to setup and execute the tools.

These metrics should accentuate the divergencies between approaches. The run time metric should be observed carefully, since it measures absolute time as opposed to computational complexity. This means that it can change between hardware used, but, more importantly, should be only compared for contemporary tools, given the increases in computing capacities. Both measures are sometimes approximate, intended to provide only the order of magnitude.

Table 2.2 - Classification of previous works.

Name	Discovery	Main Algorithm	Topology View	Evaluation	Objectives	Design Stages	Number of Topologies	Setup/Run Time
IDAC[19]	Selection	Manual	Flat	Equation	Constraint Attainment	Independent, Cell and Architecture	40	Months/few seconds
OASYS [21]	Selection	Manual Knowledge-based	Hierarchical	Equation	Constraint Attainment	Joint, Cell	>200	6 months/3 seconds
BLADES[34]	Selection	Manual Knowledge-based	Hierarchical	Equation	Constraint Attainment	Joint, Cell	NA	Long/20 min
OPASYN [22]	Selection	Manual Knowledge-Based	Flat	Equation	Overall Performance	Independent, Cell, Layout	5	2 weeks/5 min
CAMP[35]	Selection	Manual Knowledge-Based	Hierarchichal	Simulation	Constraint Attainment	Joint, Cell and Layout	NA	NA/NA
SEAS [36]	Selection	Manual Knowledge-Based	Hierarchical	Equation	Overall Performance	Independent, Cell	NA	NA/NA
Chang H A [37]	Selection	Manual Knowledge-Based	Flat	Equation	Constraint Attainment	Independent, System, Architecture, Cell, Layout (optional)	>5	NA/ >1hour

MINLP [23]	Selection	Manual/Automatic Knowledge-based	Flat	Equation	Fixed Single (Area)	Joint, Cell	64	6 months/1 min
J. B. Grimbleby, [38]	Generation	GA	Flat	Equation	Mutable Single	Joint, Cell	NA	NA/4-8 hours
DARWIN [39]	Generation	GA/ Knowledge-Based	Hierarchical	Equation	Fixed Single (Power Dissipation)	Joint, Cell	24	NA /Few minutes
FASY [40]	Generation	Automatic/Manual Knowledge Based	Flat	Equation	Mutable Single (User defined cost function)	Joint, Cell	NA	NA/NA
Koza 1997 [41]	Generation	GA	Flat	Simulation	Mutable Single	Joint, Cell	NA	NA
ASIMOV [42]	Selection	Manual Knowledge-Based	Flat	Equation	Overall Performance	Independent, Cell	4	NA
Lohn and Columbano [25]	Generation	GA	Flat	Simulation	Mutable Single Objective	Joint, Cell	NA	NA/NA
AMGIE [43]	Selection	Automatic Knowledge-Based	Flat	Equation	Overall Performance	Independent, Architecture, Cell, Layout	NA	8hr per circuit/20 min

Sripramong [17]	Generation	GP	Flat (has two transistor blocks)	Simulation	Overall Performance	Joint, Architecture, Cell, Layout	NA	NA/3 days
Dastidar [18]	Generation	GA	Hierarchical or Flat (either or)	Simulation	Overall Performance	Independent, Cell	NA	NA/1-8hr
MOJITO [16]	Generation	NSGA-II	Hierarchical	Simulation	Multi-Objective	Joint, Cell	>100,000	NA/7 days
FEATS[44]	Generation	Automatic Knowledge-based	Hierarchical	Equation-Based	Constraint Attainment	Independent, Cell, Layout	Up to 1422	NA/Several hours
Gerlach et al. [45]	Selection	Manual Knowledge-based	Flat	Equation-based	Constraint Attainment	Independent, Cell	24	NA/11 minutes
InnovA[33]	Hybrid	Causal/ Automatic Knowledge-Based	Conceptual	Simulation	Constraint Attainment	Joint, Cell	NA	NA/Several hours

2.4 Work Proposal

Given the existing works for topological synthesis it was decided to conceive a methodology that represented a novel approach. This methodology is to be used to complement AIDA-C, but its principles can be extrapolated for programs that operate in a similar way. It is possible to classify the implementation into AIDA-TOP as a topology selection tool, with flat view, using previous simulations to improve multiple objectives simultaneously, and its execution producing cell design independent from AIDA-C.

Concerning the main algorithm employed, it can be defined as knowledge-based, incorporating a different strategy from those previously observed. It relies solely on the results of stored optimizations to, from the results of their objectives, supply which topologies are more appropriate to perform as requested. This approach permits for the methodology to provide a lot of flexibility, allowing each user to store and search only optimizations of circuits, technologies and measurements that it considers valuable. Furthermore, there is no need for intricate models that require a large time investment to integrate, nor does it need long execution times to select a topology since the data it requires was previously saved and the algorithms used are simple.

The user is required only to insert the metrics to be considered. From there a series of operations are applied on the data stored within its library, resulting in a list of topologies that give him an approximate idea of those that present more potential.

3 System Architecture

3.1 Introduction

To tackle the described problem of topological creation, a methodology was created to recommend the most likely, from a set of topologies, to succeed in behaving within the desired parameters. The model extrapolates, from previously ran AIDA-C optimizations, a ranking of topologies from the most to the least fitting to attain asked specifications. The model concentrates exclusively on the objectives set by the user to derive its conclusions, disregarding the rest of the parameters.

To arrive at this ranking of topologies, a series of modules were designed. A set of AIDA-C optimizations, with the netlists and technology files it used, also have to be available (library) and the user must provide which are the relevant metrics and respective values desired (objectives). This architecture can be seen in Figure 1.1. In this section the constitution of the Topology Selection Modules and Topology Library is seen, explaining what elements they contain and what they do.

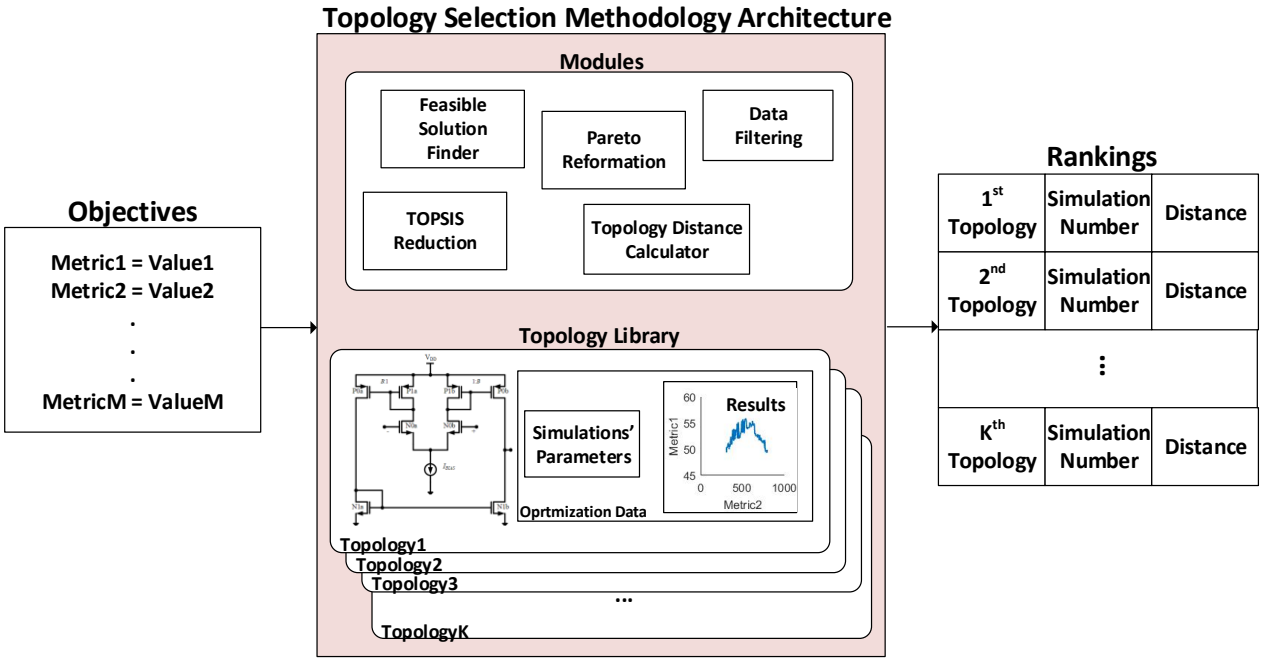

Figure 3.1- Topology Selection Methodology Architecture.

The user inputs and library data go through the modules in the sequence detailed in Figure 3.2 to deliver the output rankings. This sequence starts by firstly filtering the optimizations' data in the library to remove parameters not regarded as objectives and circuits that do not contain data for all the objective's metrics (Data Filtering module). Then, it encounters the Feasible Solution Finder, where there's a search to get points complying with objectives' specifications. If solutions are found, then only those are considered in the remaining modules, otherwise all library points are used.

The program resumes moving to the Pareto Reformation module, getting the topology optimizations' results to pareto front form. From there, for each topology it is found its extreme and compromise points through the application of Technique For Order of Preference by Similarity to Ideal Solution (TOPSIS). There are then two Topology Distance Calculators. The Closest and the Farthest Topology Distance Calculator.

The first is used when there are no feasible solutions, the Farthest if there are. In the Closest Distance Calculator, line segments between the TOPSIS module points are created. The distance from the goal to all line segments is computed. Each topology's closest point, from those that that existed in the reformed pareto, is kept, associating to it the line segment distance value. Then the points are sorted from closest to farthest. In the latter calculator the process of adding line segments is not done, only computing and storing every topology's farthest point sorted by decreasing distance.

The reason for having opposite criteria in these calculators is that if the search for feasible solutions was successful, the goal is to deliver topologies sorted by their ease in reaching the objectives, represented by larger distances to them. The topology pertaining to the farthest point from the objective is output first, and for all elements the respective distance and simulation number are provided as well. On the other occasion, the aim is to suggest, from the library, the list of topologies starting with most suited to be optimized or tweaked to fulfill all goals, ending with the least likely. The closest topology is then hypothesized as most suitable, because it will require less improvement. In this case the order is reversed still indicating the configuration that is more prone to achieve the intended results.

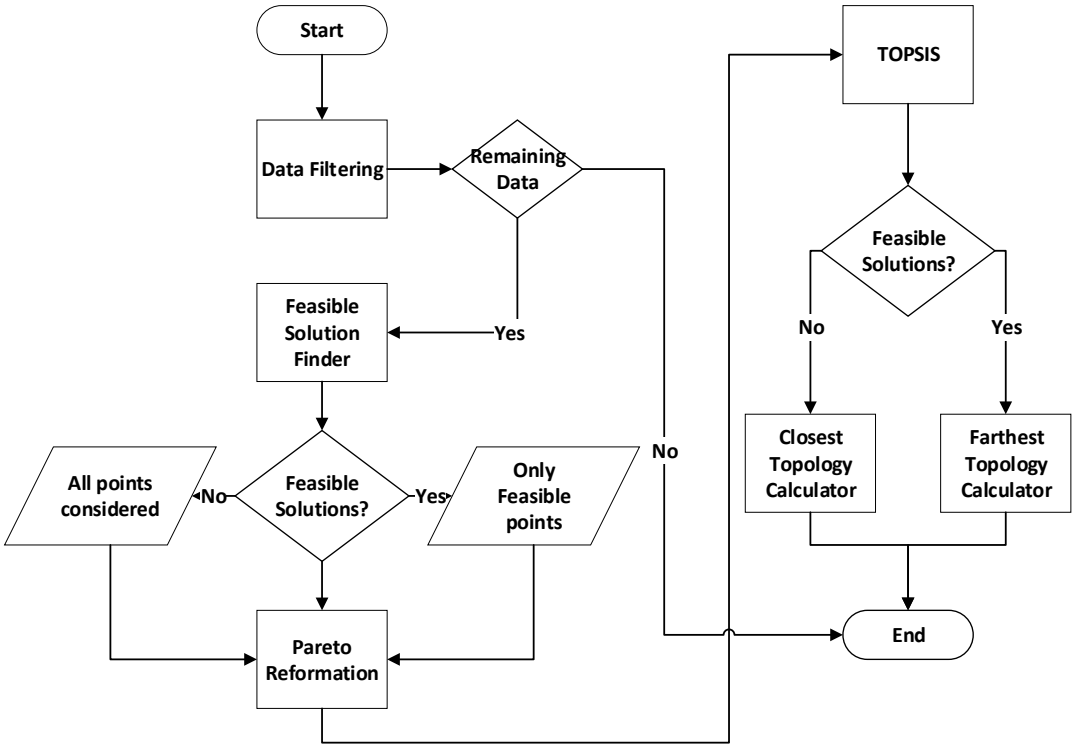

Figure 3.2 - Module sequence in Topology Selection Methodology.

3.2 Topology library

Since the premise of this work is to select a topology from a set of topologies present in a library. Thus, it is essential to define a library. In this subsection the information required for it explored, setting its guidelines, which can be effected using different implementations. Here is the essential data it must contain to allow for the application of the topology selection method. Two components are required in a library element:

- Definition of a topology.
- Results from the topology's optimizations.

The topological definition must contain information of what devices are used, technology, their respective connections. When it comes to the results from the optimizations, they are made up of several simulations. For each of them, the performance of chosen objectives is recorded together with the sizing of electrical components that led to them.

While the modules that perform the selection only use the objective performances to infer a solution, the other elements in the library are also needed. The solution is useless if it is not possible to replicate, and that is where the sizing and topological definition come in. These two together provide all necessary details for recreating, as well as altering (if necessary or desired) the selected solution, making it possible to take advantage of the output provided.

3.3 Modules

Here, the modules in this topology selection methodology are analyzed, stating their steps and purpose within the process. When useful, graphical examples are provided to make the progression through the program more intuitive.

3.3.1 Relevant Data Filter

In this module, goal parameters of each topology's optimization in the library are compared with the user asked ones. Should there be any optimizations that contain them all, then the execution continues, excluding the topologies that do not, and keeping the ones that do, limiting them to have only the input parameters. Otherwise, there is no information in the library that can aid the user's decision and the methodology terminates, declaring that there is no relevant data to help the designer.

In Figure 3.3, there are two simple examples of the operation done in this module, on a library with two topologies. The first example shows several different situations. Optimization1 on the first topology matches the input parameters prompting no filtering from the module. The second optimization on the same topology exhibits the case where one of there is an excess of information, containing the third metric that is discarded. On Topology2 no optimization contains both asked metrics, being discarded. In the second example, a case is shown where there are no elements of the library containing data on the objectives, thus leading to the termination of the execution.

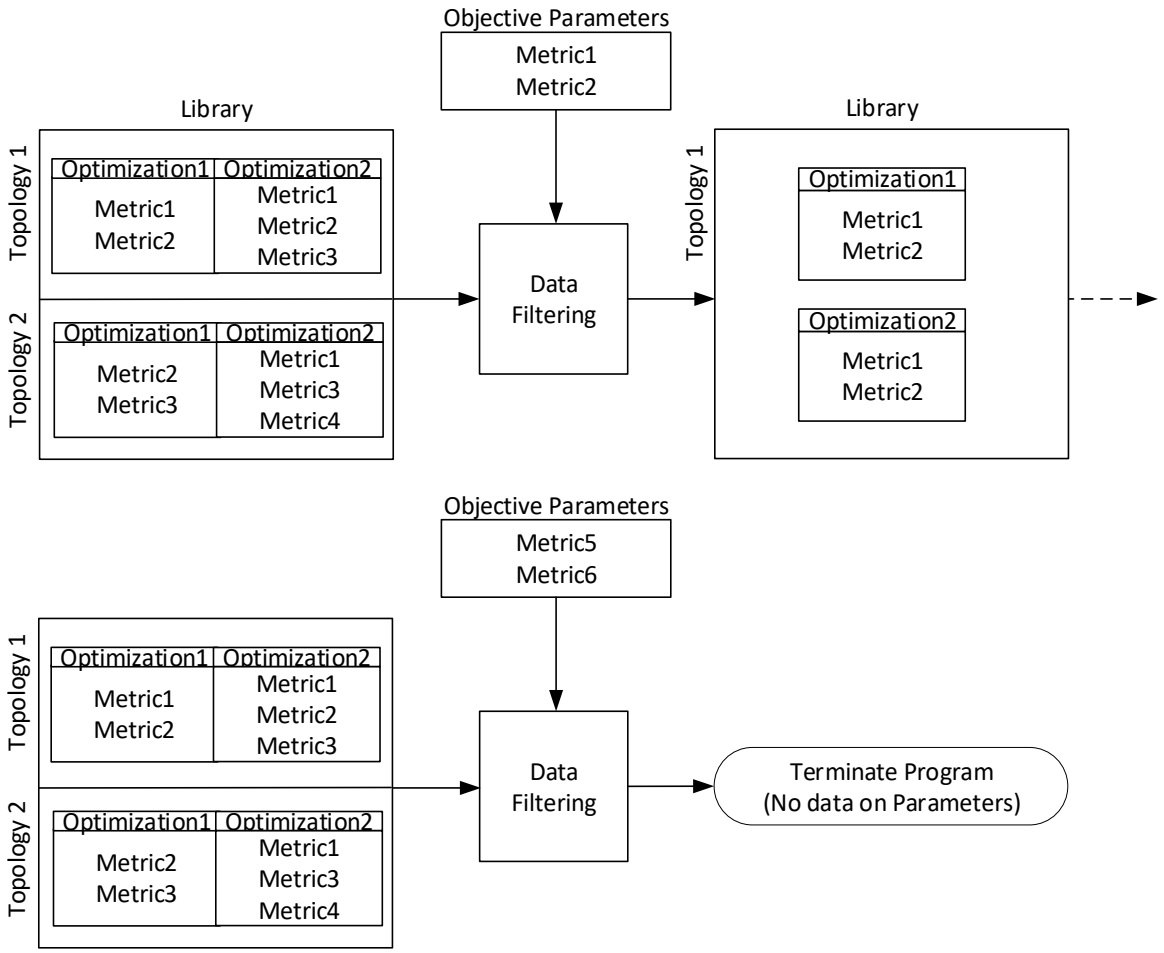

Figure 3.3 - Data Filtering Module.

This initial verification has the purpose of easily checking if there are any topologies that could potentially help in Cell Design. It can be done quickly and inform the user in case of not having useful content in the library. Furthermore, by leaving only the measurements that need to be reached, the program is freeing memory and saving on computation time that would be spent on these points.

3.3.2 Feasible Solution Finder

This part of the program goes through all simulations within the optimizations, comparing the simulation values to the objectives. If a simulation is better in all these metrics it is deemed a feasible solution and it is stored, if it is not, it is discarded. In Figure 3.4, three examples of the action of this module can be seen. On the leftmost example the asked specifications are smaller than all simulation points. Since it is desired to maximize both parameters (as in all examples) then, every point is better, thus remaining. For “Objective3” the opposite occurs where the entirety of simulations performed worse, leading to their exclusion. In the middle, “Optimization2” results are all worse in at least one component, and only a portion of “Optimization1” simulations outperform in both metrics, with the remaining fail to accomplish the asked values on the first metric, leading to their deletion.

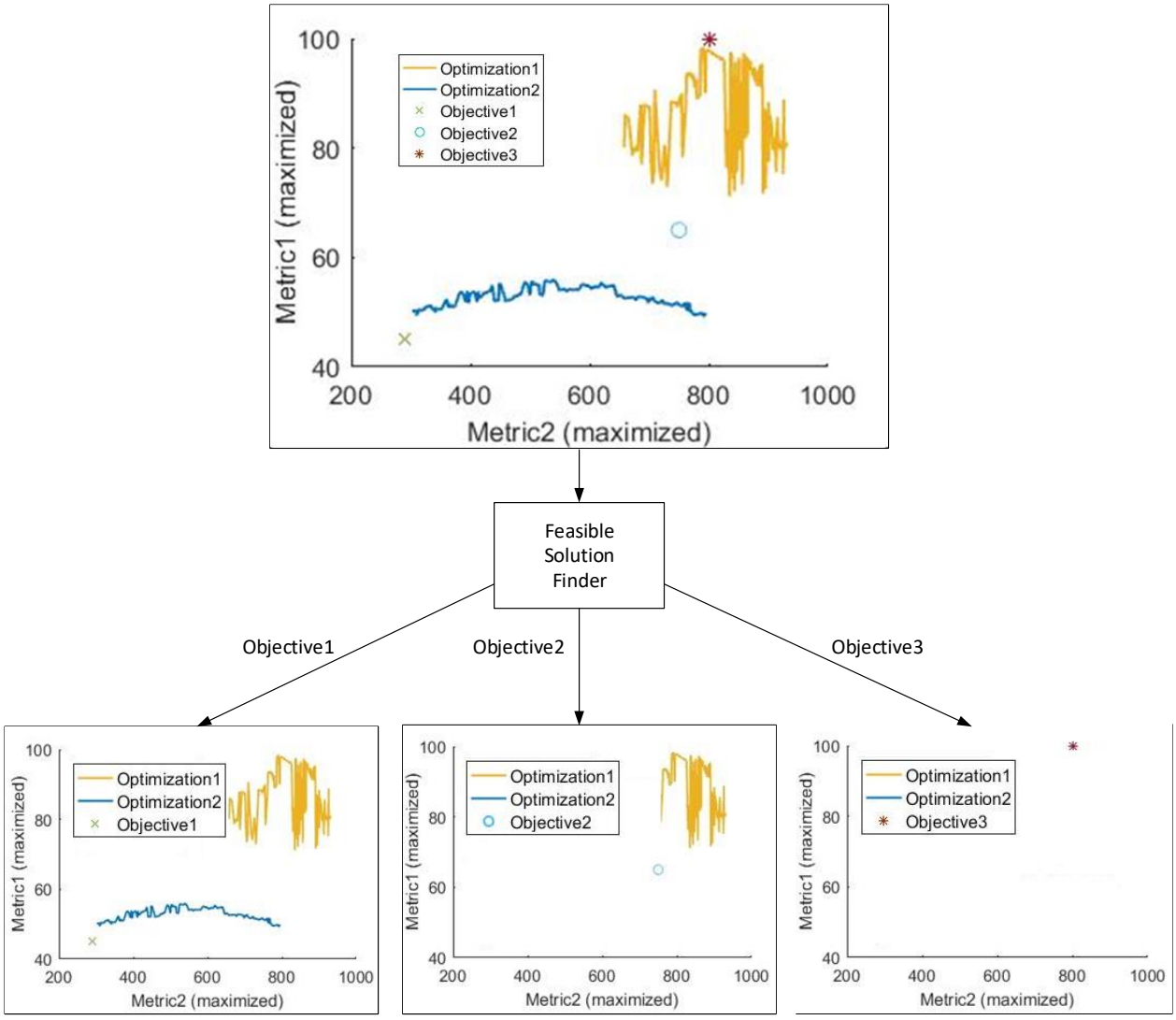

Figure 3.4 – Feasible Solution Finder module examples.

In this module the number of points decreases, bringing the same advantages explained in the previous segment, of memory and computation savings. By the end of this part it is known if there are any feasible solutions, informing the decision of which further steps to follow. If no feasible solutions are found from this step onwards the whole of set points present after the Data Filtering Module are considered. If the search encounters feasible solutions, then only these are passed on.

3.3.3 Pareto Reformation

The removal of irrelevant measurements that took place previously, may have left some dominated points. In the Pareto Reformation component, the data is restructured so that each topology contains only non-dominated points, leaving only a pareto set. To identify from the non-dominated points, the algorithm in Table 3.1 is used. It obtains non-dominated set P' from set P, which has N elements. This algorithm considers all elements of P as non-dominated at the start, copying them to the non-dominated set P'. It then iterates through all the points i in P. If P_i is still present in the non-dominated set P', it must be compared with all the points that succeed it in P (represented by j), to confirm its non-domination. In this comparison, should either of the points reveal itself as dominated (i or j), then it is removed from P', continuing to the next element of P. In the end P' contains a POF.

Table 3.1 - Pareto reformation algorithm.

$P' = P$	Copy all solution set P to set P'
$i = 1$	Set solution iterator $i = 1$ in set P
Until $i \leq N$	While there are pareto points to be evaluated
If $P_i \in P'$	If solution i of P still exists in the non-dominated set
$j = i + 1$	Set solution iterator j in set P for the point following i
Until $j \leq N$	While solution is not compared to all following solutions of P
If P_i dominates P_j	
$P' = P' \backslash \{P_j\}$	Remove j solution from P'
$j = j + 1$	Next solution of P'
Else if P_j dominates P_i	
$P' = P' \backslash \{P_i\}$	Remove solution i from P'
$i = i + 1$	Go to see if next solution of P is nondominated
$j = N + 1$	No need to compare with further elements since it is removed
Else	Solution was previously deemed dominated
$i = i + 1$	Go to see if next solution of P is nondominated

Figure 3.5 shows the results from the independent application of this algorithm on two sets of simulations results of different topologies, leaving in each the non-dominated points. If they both belonged to the same topology, all points from Optimization2 would be deleted since its results are dominated by those from Optimization1. In the first plot one can see dominated points in both lines, demarked by the circles. The connection of these points forms the pareto front seen after the Pareto Reformation module.

This algorithm is very similar to the second approach shown in the chapter from the book by Deb et al. [29], and has similar complexity. In the worst-case the number of solutions compared will be $\mathrm{N} + (\mathrm{N}-1) + (\mathrm{N}-2) \ldots + 1$ totaling $\mathrm{N}(\mathrm{N}-1)/2$ checks. Since in each check all M objectives are compared for all K topologies the complexity can be represented as $\mathrm{O}(\mathrm{KMN}^2)$. This module further restricts the points that are considered, ignoring the ones that are dominated, representing worse option.

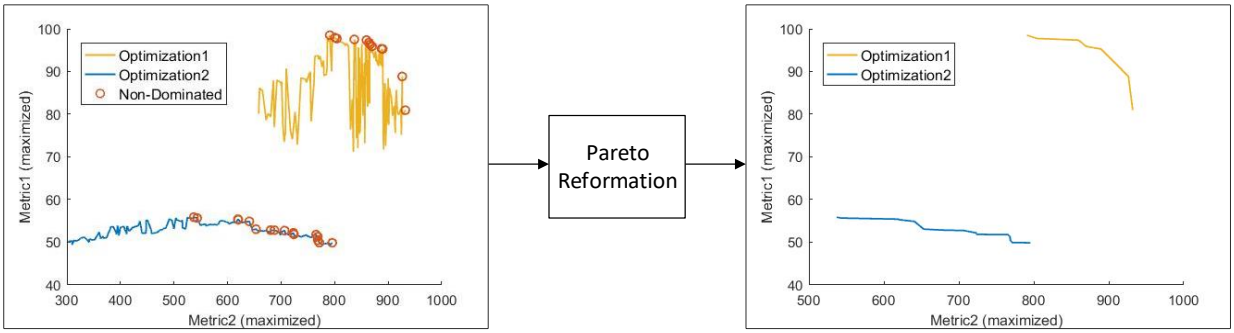

Figure 3.5 - Pareto Reformation Module.

3.3.4 TOPSIS

TOPSIS is a MCDM method (much like pareto fronts) presented by Yoon and Hwang, 1981 [46]. Instead of basing itself on the concept of domination, it uses as criteria the distance to an ideal solution. An ideal solution is one that combines the best performing elements in each of the existing dimensions. In their formulation these points are those that have maximum benefits and minimum costs. From the perspective of circuit design the benefits are the metrics that one wants to see maximized, contrary to the costs which are the specifications that must be minimized.

An ideal point can be defined as:

$$A^* = \left\{ \left(\max_i x_{ij} \,\middle|\, j \in J^* \right), \left(\min_i x_{ij} \,\middle|\, j \in J^- \right) \mid i = 1,2, \dots N \right\} \tag{3.1}$$

i as the solution x number from N available. Opposite to this the negative ideal point is:

$$A^- = \left\{ \left(\min_i x_{ij} \,\middle|\, j \in J^* \right), \left(\max_i x_{ij} \,\middle|\, j \in J^- \right) \mid i = 1,2, \dots N \right\} \tag{3.2}$$

with the same variables as before. In these points from M total criteria there is:

$$J^* = \{ j = 1,2, \dots s \mid j \; represents \; benefits \; criteria \}$$
$$J^- = \{ j = 1,2, \dots t \mid j \; represents \; costs \; criteria \}$$

In this algorithm the choice is made considering the distance from each solution to both these points, using closeness metric:

$$C_i = \frac{d_{iA^-}}{d_{iA^-} + d_{iA^*}}, i = 1,2, \dots, N \tag{3.3}$$

Where d_{ab} stands for the Euclidean distance between generic M dimensional points a and b:

$$d_{ab} = \sqrt{\sum_{j=1}^{N} \left(a_j - b_j \right)^2} \tag{3.4}$$

The closeness factor stands between 0 and 1, and the largest of computed factors in all points indicates the closest to the ideal point and farthest from the negative ideal, thus being the chosen. This is named the compromise point.

Due to the potential discrepancy in orders of magnitude on the criterions used, the distances require normalization. Normalization is applied to the initial solutions p_i from pareto P' by dividing all elements' components j by the respective sum square roots across all p_i solutions.

$$x_{ij} = \frac{p_{ij}}{\sqrt{\sum_{i=1}^{N} p_{ij}}} \tag{3.5}$$

The TOPSIS method usually also has a weighting factor, depending on preferences towards of the objectives. In the methodology presented here, it is assumed no preference, saving the one best performing in terms of closeness (compromise), additionally storing the extremes, which are the points who best perform in a single objective.

The steps to arrive at these ultimate sets are described in the Table 3.2 creating the TOPSIS routine. This is applied to all K topologies from the ones left. To start, the points that have the best performance in the jth objective ($j = 1,2, \dots M$) are saved into reduced set R, at the same time calculating the jth coordinate of ideal and anti-ideal points A_j^*/A_j^-, and this coordinate's normalization factor $normFact_j$. Upon conclusion, this normalization factor is applied iteratively to jth coordinate of the anti-ideal and ideal-points, and to all N pareto points p_i. Having all normalized elements, the distances from ith point to ideal and anti-ideal points are calculated, as is the respective closeness factor. Finally, the point with the compromise C_i, maximum closeness, is saved into reduced pareto R. The complexity, per topology, of this algorithm for the is $O(MN)$.

Table 3.2 - TOPSIS algorithm.

Until $j \leq M$	Until there are no more dimensions
$R = R \cup \{arg \operatorname{best}_i p_{ij}\}$	Add dimensional best to reduced set
$A_j^* = \left(\max_i x_{ij} \middle\| j \in J^*\right), \left(\min_i x_{ij} \middle\| j \in J^-\right)$	jth coordinate of the ideal point
$A_j^- = \left(\min_i x_{ij} \middle\| j \in J^*\right), \left(\max_i x_{ij} \middle\| j \in J^-\right)$	jth coordinate of the Negative ideal point
$normFact_j = \texttt{root\_square\_sum}(p_{ij})$	jth coordinate of the normalization factor
$j = j + 1$	Next dimension
Until $j \leq M$	Until there are no more dimensions
$A_j^* = A_j^*/normFact_j$	Normalization of the ideal point
$A_j^- = A_j^-/normFact_j$	Normalization of the negative ideal point
$j = j + 1$	
Until $i \leq N$	Until all points have been considered
$x_{ij} = p_{ij}/normFact_j$	Normalization of pareto points
Until $i \leq N$	Until all points have been considered
$d_{iA^*} = \texttt{euclidian\_distance}(x_i, A^*)$	Calculate Euclidean distances d_{iA^*}
$d_{iA^-} = \texttt{euclidian\_distance}(x_i, A^-)$	Calculate Euclidean distances d_{iA^-}
$C_i = \texttt{closeness}(A^*, A^-)$	Get the closeness C_i
$i = i + 1$	Next point
$R \cup \{arg \max_i C_i\}$	Add point with maximum C_i to set R

In Figure 3.6 it is seen a representation of the ideal and anti-ideal points, as well as the compromise and extreme ones obtained from the depicted pareto. These points are used to help in making linear approximations in the Closest and Farthest Topology Calculator.

3.3.5 Closest Topology Calculator

This module is used for the finding the closest topology in the event of not having found any feasible solutions. To get the preferred solutions, a distance-based metric is used. The metric in question is based on the Euclidean distance (visible in equation (3.4)) from the goal to line segments which are obtained through connecting the dimensional extremes to compromise point. All these points are contained in reduced pareto R (obtained in TOPSIS). The line segments represent a linear approximation of the performance that a topology can realize.

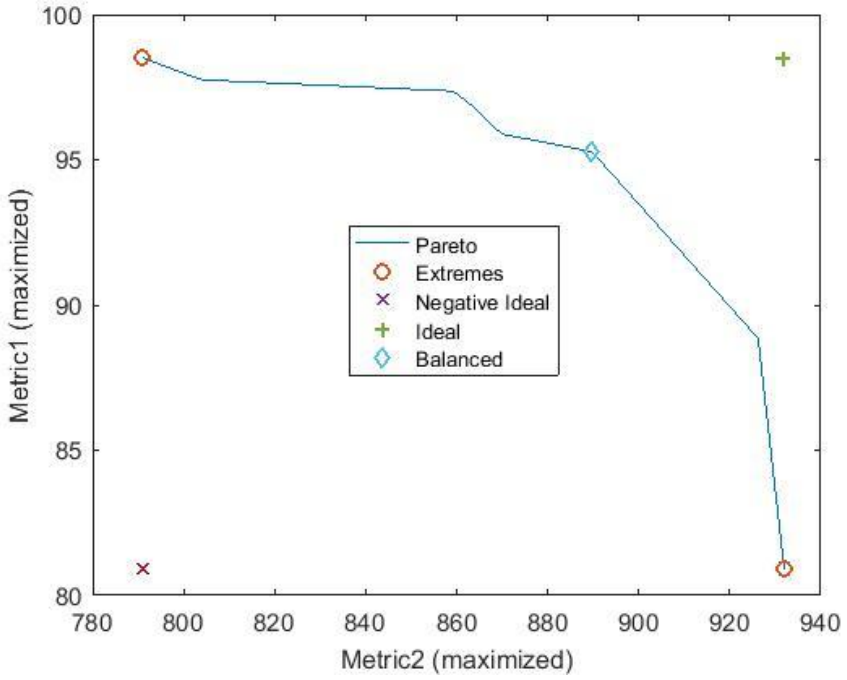

Figure 3.6 - Example of the points selected by the TOPSIS algorithm in a pareto, which will form the reduced pareto.

Due to the multi-dimensional nature of the problem, the general vectoral procedure for line-point distance computation was chosen. This method is more easily understood graphically. Let us consider that there is a line segment l vector $\overrightarrow{ab}$, covering the whole of the line segment that goes from a to b, $a, b \in \mathbb{R}^n, \forall n \in \mathbb{N}$. It is necessary to find the distance to target point t. The vector $\overrightarrow{at}$ goes from point a to the target point. The line-point distance is the norm of the vector orthogonal to $\overrightarrow{ct}$ going from l's closest point to target c, and target point t. An example of this geometric scheme can be seen in Figure 3.7.

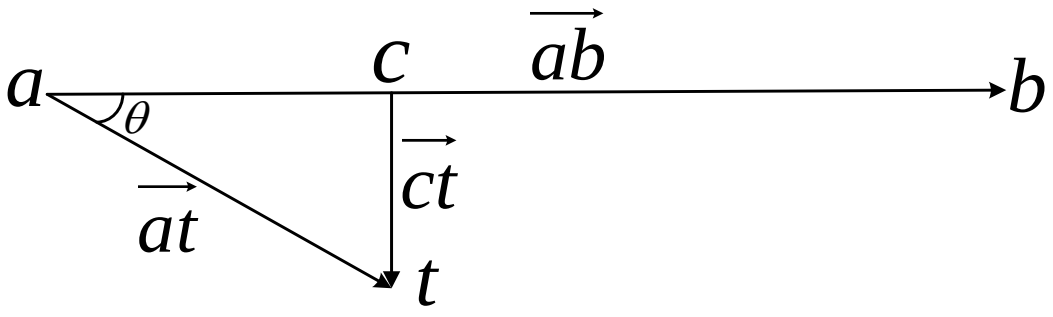

Figure 3.7 – Geometric scheme of line-point distance.

The coordinates of c can be extracted from:

$$c = a + k \cdot \overrightarrow{ab} \tag{3.6}$$

With,

$$k = \frac{|\overrightarrow{ac}|}{|\overrightarrow{ab}|} = \frac{|\overrightarrow{at}|\, cos(\theta)}{|\overrightarrow{ab}|} = \frac{\overrightarrow{(at} \cdot \overrightarrow{ab)}}{|\overrightarrow{ab}|^2} \tag{3.7}$$

The point c however only belongs to the line segment if,

$$0 \leq\ k \leq\ 1 \tag{3.8}$$

Observable by noting that the definition of c commences in the a extreme of the line segment, and that the $\overrightarrow{ab}$ vector multiplying by k reaches the l's other extreme, b. For k obeying to equation (3.8), c is valid, being trivial to get $\overrightarrow{ct}$ whose norm, represents the point-line distance. If k is outside this boundary, the vector of both ends (a and b), to t with smaller norm is the one whose norm represents the distance from the line segment to target.

To apply this computation of line-point distance, first it is necessary to extract the vectors defining the lines in which the line segments are included. This vectorization is made by subtracting the extreme points with the balanced, as per the definition of a vector. Then the same is done between the extreme point and the objective point. Now, for all line segments, it is possible to obtain its k using (3.7), saving the points that respect (3.8), with the small modification of not including the points at both ends, since they are already accounted for.

This action is performed for the wealth competing topologies, subsequently measuring all distances between the goal point and the reformed pareto P' (result pareto reformation module), plus the newly added closest points within the line-segments. All the initial points used in this part have been subject to the normalization occurring in the subsection 3.3.4, thus presenting the distances balanced by their range of values. Since what is wanted if feasible solutions weren't found is the closest point, the results are presented with ascending distance.

If an added point is the best within a topology, the closest pre-existing point from Pareto set P' from must be indicated. These added points only serve to supply a more useful distance metric, not having associated any actual replicable device sizing. Due to this, the distance to this point is registered (if best), as the distance to the topology itself, yet the simulation that the program outputs has to contain a specification, allowing the user to work from it. Therefore, the distance value to all topologies is sorted using both added points and simulation-based ones (P'points), but once the topology ordering is achieved, the topologies' nearest simulated point is singled out, to be displayed in the output.

Table 3.3 - Topologies distance sorting algorithm.

Until $j < K$	Until there are no more topologies
$\overrightarrow{bt} = t - x_M$	Compromise to target vector
Until $i < M$	Until all line-segments are considered
$\overrightarrow{bx_\iota} = x_i - x_M$	Line-segment vector from compromise x_M to extreme x_i
$k =$`get_k`$(\overrightarrow{bx_\iota}, \overrightarrow{bt})$	Application of equation (3.7)
If $0 < k < 1$	Check if closest point c_i will belong in the line segment
$c_i = b + k \cdot \overrightarrow{bx_\iota}$	Computation closest point go target c_i
$P'_j = P'_j \cup \{c_i\}$	Addition of c_i to set of pareto points of topology j, P'_j
$D_j =$ `distances`(P'_j, t)	Get distance from points in P'_j to t
$g = \arg\min_{\mathrm{j}} D_j$	Find smallest element in D_j
$S_j = (\min D_j, P'_g, j)$	S_j with smallest distance, correspondent element and topology
$S = S \cup \{S_j\}$	Set with selected points S_j for competing topologies
$i = i + 1$	Next line-segment
$j = j + 1$	Next topology
$S =$`sort`(S)	Sort selected points by distance

The distance used was said to be Euclidian distance based, but it requires a modification. Since this module mirrors the improvements needed so that all objectives are compliant, if in some of the metrics the performance is already adequate, no effort is required in them. So, the modification is that only the distances of the non-attaining objectives, contributing to the overall distance, since the other components are already ensured. It is the one with smaller cumulative distance in normalized non-conforming specifications that needs to be improved less than the others.

This module provides an inexpensive linear approximation of the circuits' performance and then finally calculates and sorts the best topologies, that will be output to the user.

3.3.6 Farthest Topology Calculator

When one stands before a feasible solution, the one that is overall farthest is said best. It is assumed to allow more changes in the remaining specifications than the rest of the points. Since the farthest point in a line-segment always lies in its ends, then only the Reduced Pareto R(set obtained after TOPSIS) is used in this module, not the whole reformed pareto, to get the best of feasible solutions.

The equation used for the distance computation is the same as in the Closest Topology Calculator. Nevertheless, in this module no points are added, instead directly calculating the distance from the target point to the reduced pareto resultant from subsection 3.3.4. After obtaining the distances to these points, feasible solutions are sorted by descending distance, opposite to the other calculator.

4 Test Library

4.1 Introduction

In the previous section the methodology to apply was explored, giving the essential steps to replicate it. To assess the usefulness of the methodology, it must be tested, resorting to specific electronic problems.

A set of circuits, constraints, and objective specifications need to be chosen, so it is possible to optimize them, resulting in data for assessing the method's success. In this section it is overviewed the elements chosen, as well as their expected behavior. Then the implementation of the library follows, consisting on how these elements were encoded to permit their optimization. Then, the format of the optimizations' output and the process undertaken in them are explained. The optimizations performed are described, stating the motivations behind them and the resulting data. After this, all that was required to run the implementation of the model and validation program was ready.

4.2 Test Library content

To prove this work's concept, it was needed to elect circuits to test it in. Due to the characteristics of the method, it requires that each performance metric is quantifiable as a unit of measurement. Within this description there are vast possibilities, one of them being OpAmps.

This type of circuit is deeply linked with the appearance of ICs, after an initial stage of fabrication with discrete components, where analog computation and sophisticated instrumentation were performed. The IC and OpAmp popularity drove their price down, quickly reaching prices around the tens of cents. This popularity stemmed from the IC OpAmp's behavior similarity to what was theorized (ideal), besides its adaptability to different uses.[47]

A related set of circuits are OTAs, whose configuration largely resembles the one in OpAmps, missing only the output stage. This modification turns it into a voltage-controlled current source amplifier, as opposed to the OpAmp, which is a voltage-controlled voltage source. They have, in the last decades, gained prominence in multiple areas in filtering and signal processing. There are several variations of OTAs, whose performance differences are widely known, given the attractiveness of this type of circuits. Moreover, these topologies are often less intricate than the operational amplifiers, simplifying the simulation process. [48] The conjunction of these last two properties make them good choices for trying the proposed method.

The OTAs are implemented with CMOS technology, since, even though bipolar transistors offer several advantages, CMOS circuits dominate markets due to their lower power dissipation and smaller size. Such factors became especially relevant with IoT, wireless and portable systems. [49]

Related to the choice of circuits, is the choice of what measures to use. If all circuits used are too similar in respect to a measure, then the answers provided will not be as relevant, since the topologies will seem

equally likely to realize it. The performance metrics should also be representative of actual parameters that designers have interest in seeing bettered, matching the context in which such methodology could be helpful. In this sense it is reasonable to choose parameters that are not always simultaneously improved, expressive of the trade-offs for which MOO was created for. [50]

These metrics are introduced at the outset of this subsection, to then provide an improved intuition of their change from circuit to circuit. Secondly the amplifiers chosen are displayed, emphasizing how they differ regarding the parameters just exposed.

4.2.1 Measures

For testing, 4 metrics were chosen: Voltage Gain, Figure-Of-Merit (FOM), Offset Voltage (VOS) and Output Swing Voltage (OS). These metrics are briefly explained in the following paragraphs.

1) Voltage Gain

The voltage gain is defined as the ratio between the input voltage v_i and the output voltage v_o which will be delivered to the load. This metric quantifies the amplification of the input signal, which, as the name indicates, is one of the main amplifier (subsequently OTA) metrics. However, in a real amplifier the gain is not always the same, in the sense that the gain $A_v(f)$ is dependent on the considered signal frequency f. In fact, it tends to drop as the frequency in question reduces and the one of interest for these experiments is extracted in the lower bands (close to zero), being denominated as low-frequency gain and defined by:

$$A_v = \frac{v_o}{v_i}[\mathrm{V/V\ or\ dB}] \tag{4.1}$$

2) Figure-Of-Merit

The Figure-Of-Merit term is used as a number that characterizes the performance of circuits in the context of the energy-efficiency and is commonly used in the literature of this sort of topologies. It weighs the power consumption which is dependent on the amount of flowing current I_{DD} with the establishing speed of the amplifier quantified by the gain-bandwidth product GBW. The GBW, or unitary frequency of the amplifier, corresponds to the frequency value where the voltage gain of the circuit is equal to 1 V/V, i.e., 0 dB, being mathematically described as in (4.2). The FOM is, therefore, defined as the product between the GBW and the load capacitance over the current consumption, as in (4.3). It is worth mentioning that in practical terms, single-stage amplifiers, such as the ones used for proof-of-concept in this work, trade low-frequency for bandwidth for the same gain-bandwidth product, but a common trade-off appears when one confronts the energy-efficiency FOM with the low-frequency gain, since to improve the gain-bandwidth product, usually designers rely on increasing the current consumption for the same load.

$$GBW = \{f \in \mathbb{R} | A(f) = 1\}\,[\mathrm{Hz}] \tag{4.2}$$

$$FOM = \frac{GBW \times C_l}{I_{DD}}\left[\frac{MHz \times pF}{mA}\right] \tag{4.3}$$

3) Offset Voltage

The offset voltage V_{OS}, is defined, in this work, as the difference between the actual DC voltage that is applied at the output by the amplifier, and the value that would be achieved in an ideal situation (ideal amplifier), which would be half of the positive supply voltage V_{DD}. This is, therefore, a generic non-ideality or degree of imperfection of real circuits, which can be defined as in (4.4). In CMOS amplifiers, the offset voltage is a relevant measure because it may cause distortion and also cause offset cascade effects in the circuitry ahead. The nature of the offset voltage is twofold: the random offset and the systematic offset. The random offset is due to the geometrical mismatching and process dependent imprecisions. The systematic offset can be reduced to a value close to zero with a cautious design, since this measure results from the design of the circuit and is present even when all the matched devices are physically identical.

$$V_{OS} = V_o - \frac{V_{DD}}{2}\ [V] \tag{4.4}$$

.

4) Output Swing Voltage

The output swing voltage is defined as the maximum swing of the output node without generating a defined amount of harmonic distortion. In practice, the OS is determined by the difference between the positive supply and the negative supply voltages or ground minus the overdrive voltages of the transistors that drive the output node, i.e., transistors that form the output branches of the amplifier. Amplifiers have a desirable linearity, within a limited range of output voltages. If the amplified voltage output surpasses the interval of values for which the transistors of the output branch can remain in the amplification conducive mode (saturation, for MOSFET), then the signal will be distorted, clipping it for voltages after the limits have been crossed. The output swing voltage represents this interval of permitted values that should be maximized. The voltage considers only half of the signal, which through the output port symmetry, is equal to the other half. It is obtained noticing that the output voltage V_o, needs to be sufficient to ensure the saturation of the remaining n transistors in the output branch, by being superior to the sum of their saturation voltages V_{DSAT}.

$$OS = 2 \times \left(V_o - \sum_{i=1}^{n} V_{DSAT}^{i} \right) [V] \tag{4.5}$$

4.2.2 Circuits

The chosen CMOS OTA topologies are now listed. For the purposes of this work, four topologies are considered for proof-of-concept: the Symmetrical CMOS OTA, the Telescopic-Cascode OTA, the Folded-Cascode OTA and the Mirrored-Cascode OTA. The circuits are detailed in both functional and analytical contexts, focusing on the structural differences between each topology and how the changes affect the performance metrics. This way, the circuits' measures will be evaluated comparatively in a qualitative manner.

The symmetrical CMOS OTA, shown in Figure 4.1a, is the most suitable circuit if a high output swing is desired, since a minimum number of stacked transistors are employed in the output branch. This circuit is composed of a differential pair in the middle, with active PMOS loads that also operate as current mirrors for the output branch. As a traditional OTA, the input is in voltage and the output is in current, therefore only capacitive loads or very high resistive impedances can be hanged at the output. The gain of the symmetrical CMOS OTA is dependent exclusively on the transconductance of the differential-pair *gm*, on the current mirroring factor *B*, and on the output resistance r_0, as given in (4.6). Furthermore, the GBW, given in (4.7) is directly proportional to B and the first two terms and minimized by the capacitive load. As introduced, a key advantage of the symmetrical CMOS OTA is the better OS, which can be given by (4.8).

$$A_v = gm \times B \times r_0 \tag{4.6}$$

$$GBW = \frac{gm \times B}{2\pi \times C} \tag{4.7}$$

$$OS = V_{DD} - 2 \times VDS_{SAT} \tag{4.8}$$

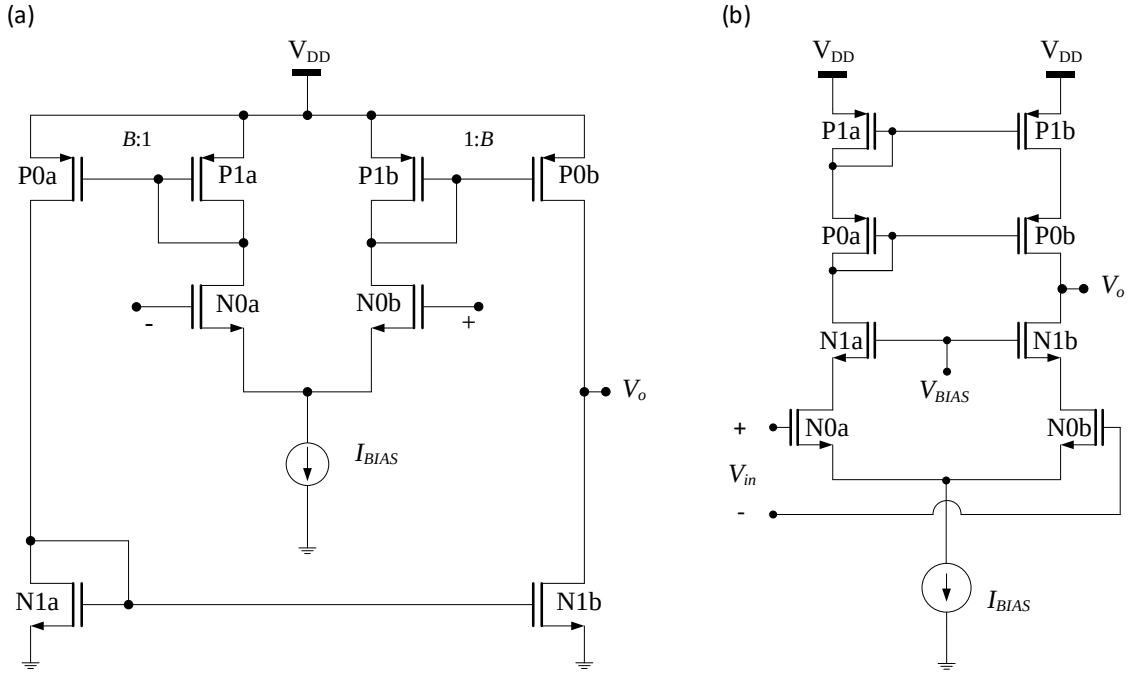

Figure 4.1 – Two CMOS OTAs chosen for testing: (a) Symmetrical (b) Telescopic-Cascode.

The Telescopic-Cascode amplifier (TCA), shown in Figure 4.1b, often has a gain higher than the symmetrical CMOS OTA and provides, in general, a good tradeoff between gain, power consumption and speed; but the output swing of this architecture is limited. The cascode designation has been brought through times since the days when scientists used vacuum tubes, and results from the grammatical contraction of two words: cascaded and cathode, hence cascode. The gain of the TCA can be given by the direct contributions of the transconductance of the differential pair and the output resistance. The gain is shown in (4.9), and the output resistance is given by (4.10). The transconductance of the amplifier is equal to the transconductance of the differential-pair, since cascading has no direct impact on the transconductance of the amplifier.

$$A_v = gm_{N0} \times \left(\frac{gm_{P0} \times gm_{N1} \times \left(r_{o_{p1}} \times r_{o_{p0}} \times r_{o_{N0}} \times r_{o_{N1}} \right)}{gm_{P0} \times r_{o_{p1}} \times r_{o_{p0}} + gm_{N1} \times r_{o_{N0}} \times r_{o_{N1}}} \right) \tag{4.9}$$

$$r_0 = \left(\left(gm_{P0} \times r_{o_{p0}} \right) \times r_{o_{p1}} \right) \parallel \left(\left(gm_{N1} \times r_{o_{N1}} \right) \times r_{o_{N0}} \right) \tag{4.10}$$

The Mirrored-Cascode amplifier (MCA), shown in Figure 4.2 can be considered as a combination of a Telescopic-Cascode amplifier with the symmetrical CMOS OTA. The MCA uses current-mirrors to drive the output node, improving the gain by the mirroring factor *B* (4.11), as in the symmetrical CMOS OTA. Only four transistors in series improve the output swing of the amplifier when compared to the telescopic approach. The major drawback of this topology, when compared to the Telescopic-Cascode amplifier is the power consumption which is higher, precisely due to the current-mirroring technique. The power dissipation is approximately equal to $V_{DD} \times 2(1+B) \times I_{BIAS}$, which is, expectedly, greater than the telescopic approach by a factor of *B*.

$$A_v \propto gm \times B \times r_0 \tag{4.11}$$

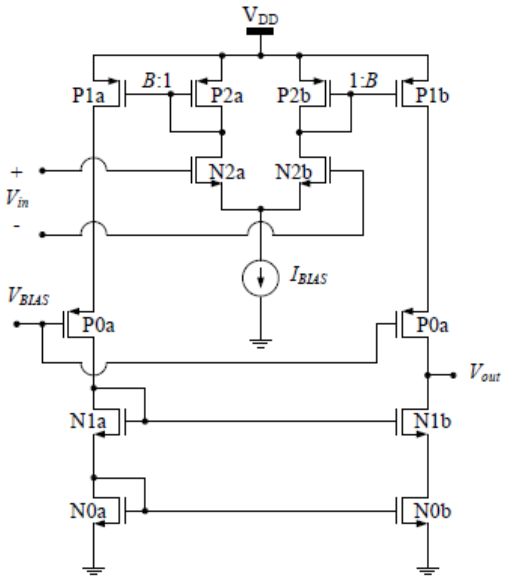

Figure 4.2 - Mirrored-Cascode Amplifier.

In order to reduce the impact of stacking a large number of transistors across a lower voltage power supply, it is possible to replace the common-gate NMOS transistor of the Telescopic-Cascode stage, N1, by a PMOS device in a common-gate configuration, i.e., the gate is common to both input and output, from the small-signal point of view, in such way that the basic behavior principles and analytics remain the same. This topology is called folded, since the PMOS reverses the direction of the signal flow back to ground. The Folded-Cascode can provide greater OS than the Telescopic-Cascode and increases the common-node input range as well, becoming more independent in terms of DC voltage. However, by using PMOS transistors, the non-dominant pole is at a lower frequency due to the fact that the PMOS have lower transit frequency, which can have implications in terms of GBW, which can be lower. The gain can be determined through (4.12), (4.13), (4.14) and is defined as in (4.15).

$$r_{oUP} = gm_{P0} \times r_{o_{P0}} \times (r_{o_{P1}} \parallel r_{o_{N2}}) \tag{4.12}$$

$$r_{oDOWN} = gm_{N1} \times r_{oN1} \times r_{oN0} \tag{4.13}$$

$$r_o = r_{o_{UP}} \parallel r_{o_{DOWN}} \tag{4.14}$$

$$A_v = gm_{N2} \times r_0 \tag{4.15}$$

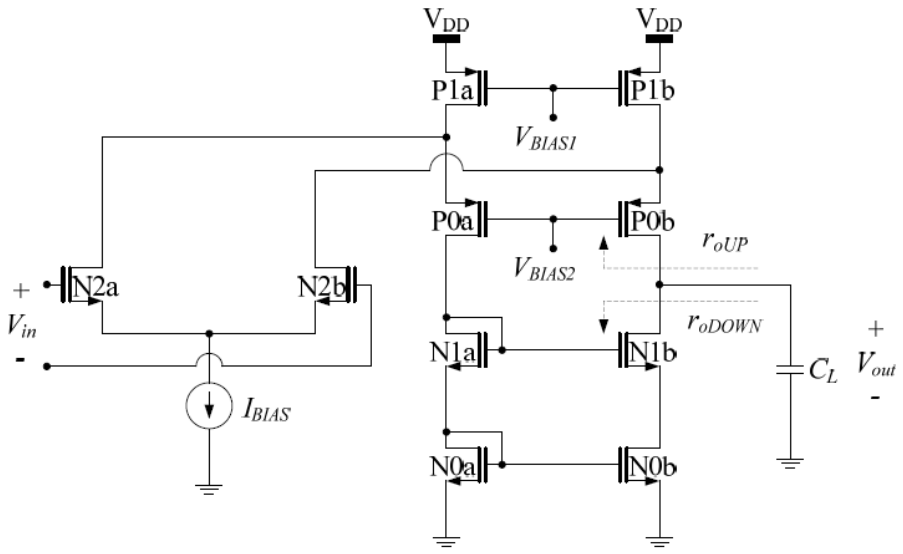

Figure 4.3 - Folded-Cascode Amplifier.

A comparison between the Symmetrical, Telescopic-Cascode, Mirrored-Cascode and Folded-Cascode amplifiers is presented in Table 2.1, summarizing the topologies used as proof-of-concept in this work. A brief summary of the maximum output swing of these circuits is presented in Table 2.2

Table 4.1 - Relative performance of circuits.

Topology	Gain	Output Swing	Speed	Power Dissipation
Symmetrical	Low	High	Low	Moderate
Telescopic-Cascode	Moderate	Low	Moderate	Low
Mirrored-Cascode	High	Moderate	Moderate	High
Folded-Cascode	High	Moderate	High	Moderate

Table 4.2 - Output signal swing expressions.

Topology	Maximum positive output signal swing $V_{o_{MAX}}^{+}$	Maximum negative output signal swing $V_{o_{MAX}}^{-}$	Maximum available output signal swing $V_{o_{MAX}}$
Symmetrical	$V_{DD} - VDS_{SAT}$	$-VDS_{SAT}$	$V_{DD} - 2 \times VDS_{SAT}$
Telescopic-Cascode	$V_{DD} - 2 \times VDS_{SAT}$	$-3 \times VDS_{SAT}$	$V_{DD} - 5 \times VDS_{SAT}$
Mirrored-Cascode	$V_{DD} - 2 \times VDS_{SAT}$	$-2 \times VDS_{SAT}$	$V_{DD} - 4 \times VDS_{SAT}$
Folded-Cascode	$V_{DD} - 2 \times VDS_{SAT}$	$-2 \times VDS_{SAT}$	$V_{DD} - 4 \times VDS_{SAT}$

4.3 Library Implementation

In subsection 4.2 it was stated the sufficient information that had to compose the library, so that this methodology would work giving general guidelines for creating one. Here it is seen how it was implemented by us.

One of the initial motivations for the realization of this project was creating an auxiliary tool for complementing the universe of AIDA software. AIDA is an EDA solution working on the Cell layout with AIDA-L and Cell Design stages of analog circuits creation. Within Cell Design, it presents only a solution relative to component sizing AIDA-C. The latter is the one that was utilized for this work.

The execution of this tool outputs an XML file, containing a pareto of circuit solutions. Each solution is made-up by the objective metrics' simulation values and the dimensions of transistors the simulation based itself to obtain them. These dimensions are named in this file according to the netlist of the optimized circuit, thus requiring it for re-creating a fully sized version of the circuit. Furthermore, the netlist of the setup for electrical testing (test bench), must also be provided so that it is possible to know how the performance metrics were measured. Such netlists were created to be used with the Eldo® simulator employed within the AIDA-C program, to assess circuit performance. There are also library technology files, which hold the transistors' mathematical model for accurate transistor behavior simulation.

The library created is a combination of the circuit and test bench netlists (one per topology mentioned), technology files, and AIDA-C XML files, each with an associated pareto. As to test the possibility of resorting to simulations with larger number of goals to predict the success on predicting for fewer goals, these simulations regarded all aforementioned performance metrics. The details of AIDA-C runs and their results are seen in subsection 4.4.4.

4.4 Optimizations

The following topics are approached in this part: AIDA-C's workflow and overview of how it acts to get the data needed, the search space and constraints set in the design file, the output file structure and how the relevant content was extracted, analysis of the library's paretos, description of the test set.

4.4.1 AIDA-C

Even though the AIDA-C tool has already been mentioned in this report, it remains to be described more precisely what it does and the steps to initiate it. It is a layout-aware tool for optimizing circuit sizing, using an altered version of the NSGA-II algorithm. The program uses this EA algorithm to search through a user defined range of transistor characteristics. The sizing process is enhanced by performing Monte Carlo yield optimization [51], along with considering process, voltage, and temperature corners (PVT). AIDA-C can resort to multiple simulators from which the user can choose one.

```
1: <?xml version="1.0" encoding="ISO-8859-1"?>
2: <!DOCTYPE Design SYSTEM "design-1.0.dtd">
3: <Design name="VoltageCombinersDiffAmplifier">
4:  <Circuit technologyNode="UMC_013">
5:    <Var id="w8 w6 w4 w10 w1 w0" range="1e-6:1e-7:1e-4"/>
6:    <Var id="nf8 nf6 ... nf0" range="1:2:8"/>
7:    <Var id="l8 l6 l4 l10 l1 l0" range="3.4e-7:1e-8:1e-6"/>
8:  </Circuit>
9:  <TestbenchSetup simulator="eldo">
10:   <TestCase name="SS"/>
        ...
16:   <NominalTb netlist="ac_testbench.cir.eldo">
17:     <Meas id="idd"  desc="Current Consumption [A]"/>
        ...
23:     <Meas id="ov_mp0 ... ov_mn11" desc="Overdrive [V]"/>
24:   </NominalTb>
25:   <WorstCaseTb netlist="ac_testbench.cir.eldo.corners">
26:     <Meas importFrom="ac_testbench.cir.eldo"/>
27:   </WorstCaseTb>
28: </TestbenchSetup>
29: <!-- Predefined constraints; can add/remove/edit in GUI-->
30: <Constraint op="GE" value="0.10" meas="vov_mp0 ..." />
     ...
39:</Design>
```

Figure 4.4 – Sample of AIDA-C input XML

To use the program, it is required that the user create a folder with circuit and test bench netlists, ensuring their compatibility with the simulator option taken. Based on this netlist an input XML file is written (example on Figure 4.4), that defines the variables for the circuit, the range of the transistors dimensions acceptable, the constraints to maintain the transistor in the required operation mode, and constraints on electrical metrics (including minimum requirements for objective metrics). It also points towards the netlists that the

simulator will use, and the library technology files. Optionally an image of the circuit is included for the designer to know what the circuit is like without having to import it into another software.

While using the AIDA-C graphical user interface (GUI) which can be seen on , the number of generations, size of population, crossover and mutation rate, and objectives to be optimized are defined, completing the preparation stage. The program is started, ultimately delivering a POF of circuit solutions, if it was able to satisfy all the constraints.

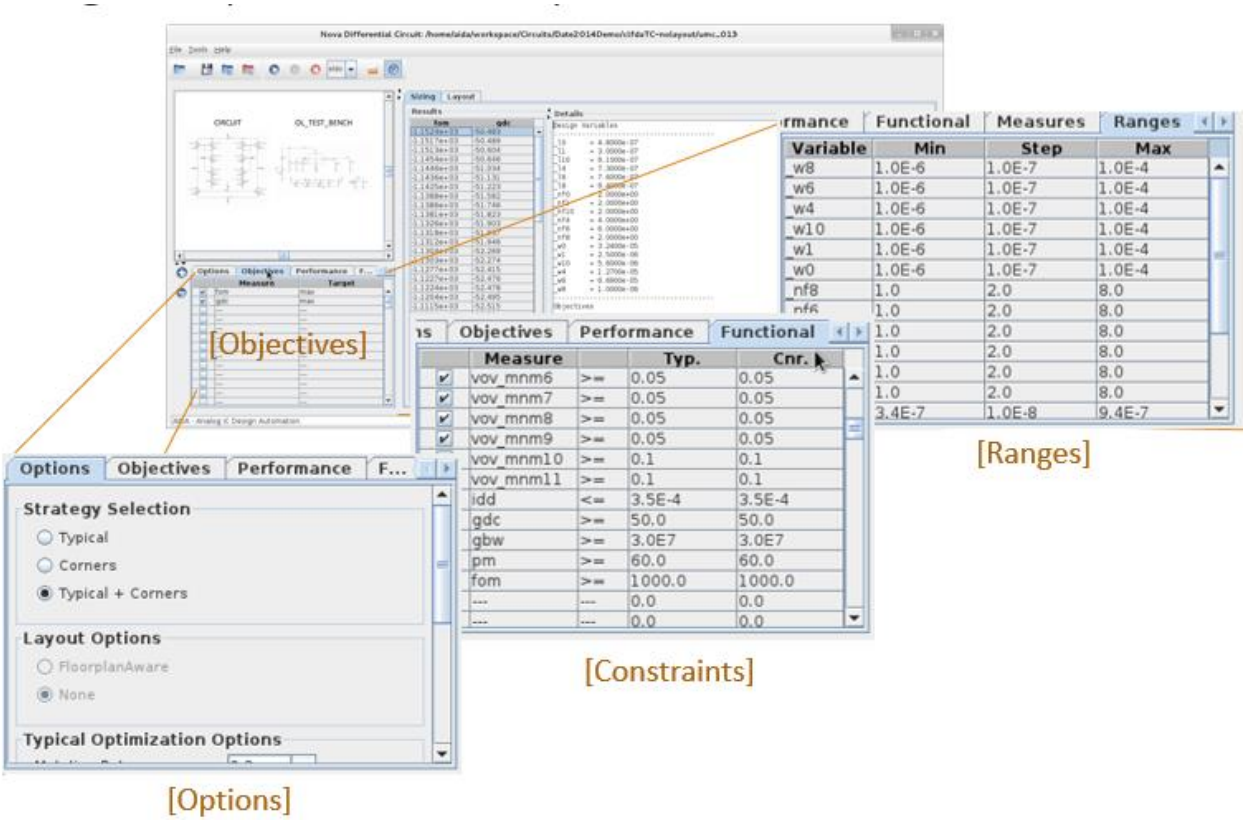

Figure 4.5 – AIDA-C GUI

4.4.2 Search Space and Constraints

The search space, since AIDA-C is a sizing tool, has as variables the measurements of the transistors that integrate a circuit. For a given transistor there are three dimensions that can be altered affecting its performance: width, length and number of transistor fingers. In the design files of the topologies it is defined the range The *Step* relays the minimum intervals that are considered in between possible dimensions. The number of possible values is:

$$\#PossibleValues = \frac{Maximum - Minimum}{Step} \tag{4.16}$$

The possibilities within a transistor are calculated by doing the product of each characteristic's possible values. Across all transistors the following values were decided, through design experience, for these dimensions:

Table 4.3 - Transistor dimension possible values.

	Minimum	Maximum	Step	#PossibleValues
Width	2	298	0.1	2960
Length	0.36	2.36	0.1	20
Number of Fingers	1	16	2	8

Some transistors must, for symmetry reasons, have identical dimensions to others. The dimension variables for the circuits are discriminated in Table 4.4.

Table 4.4 - Topology number of dimension variables

	Symmetrical	Telescopic-Cascode	Mirrored-Cascode	Folded-Cascode
Width	6	9	10	9
Length	6	9	10	9
Number of Fingers	4	7	7	6

When it comes to constraints there were also set minimum values for important specifications to which these circuits generally must assure: (1) power consumption, (2) VOS, (3) Gain, (4) GBW, (5) Phase Margin, (6) FOM. These constraints were set to the same value in all design files, varying only which of them were defined as objectives.

Finally, there are two biasing voltages that each transistor must comply with: delta voltages and overdrive voltages, to maintain the transistors in the desired operating region (saturation). The number of transistors in the topologies are: Symmetric - 10; Telescopic-Cascode - 13; Mirrored-Cascode - 16; Folded-Cascode- 14 transistors.

4.4.3 Output file

In the output file it is possible to find all the information that the topology selector needs. Additionally, it has a wealth of other data, some of which is also useful for this work. Starting with the most important elements in this file, inside the project's data lays the "Measure evaluation" object, that has in its "Objectives" field the information about the goals set for optimization. From it, it can be seen how many were the measures that the user wished to optimize, what they were, and if the optimization consists on maximizing or minimizing a metric. Of equal importance is the Field "F" within the "SizingList". This list represents the pareto, with the list elements ("SizedCircuits") being the solutions its solutions. In "F" it can be found, in the same order as it was seen in the "Objectives" field, the performance on each of the objectives. The information mentioned in this paragraph is what was used in our methodology, for which a MATLAB® script was created to retrieve and store it in a more succinct way.

For recreating the circuits, the "RangeArray" object has the variables that were varied throughout the program execution. In the same logic as with the objectives, the values that originated each pareto are displayed sorted in the "X" object contained in the "SizedCircuit" elements. In addition to this information it is possible to check the circuit's name, the files and configurations that were at its inception, the time necessary to execute, amongst other information.

4.4.4 Library Paretos

The netlists and input XML files for the circuits chosen were created. Within AIDA-C program the objectives to optimize were set to cover all the metric options presented in 4.2.1, selecting the population size (128 individuals) and performing four 1000 generation iterations making for a total of 4 thousand generations per circuit. The constraints and genetic algorithm configurations conform to those in state-of-the-art optimizations for the same technology and topologies. The option to arrest optimization at 4000 generation was based on three factors: (1) design experience, (2) by visually observing the progression of an AIDA-C plot indicating the algorithm's progress, (3) the stability of the paretos also plotted in AIDA-C optimizations. [52],[53],[54]

To represent the resulting outputs the projections of the optimizations' results were split into two pairs. The first is Gain and FOM, seen in Figure 4.6, and in Figure 4.7 VOS and OS are displayed. These pairs were chosen based on the higher connection between them.

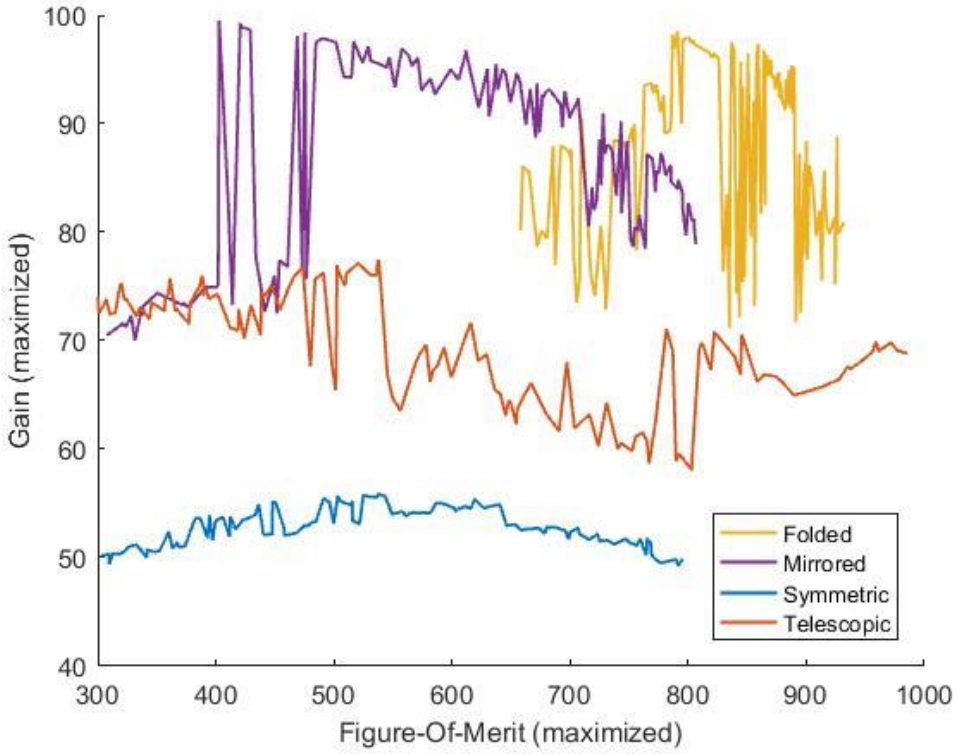

Figure 4.6 - Projection of FOM and Gain output from 4000 generations of AIDA optimizations.

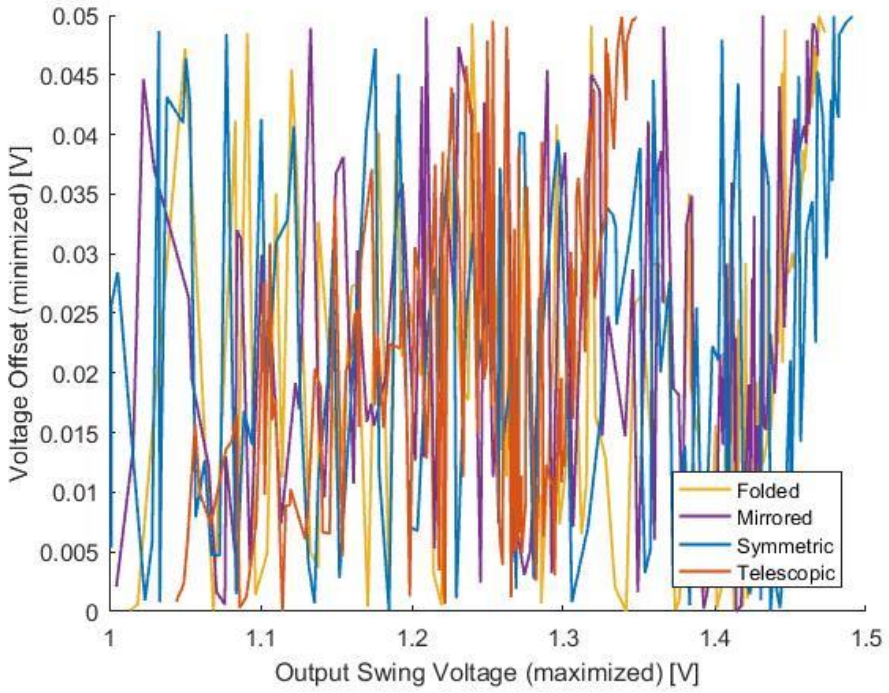

Figure 4.7 - Projection of VOS and OS output from 4000 generations of AIDA optimizations.

5 Tests and results

The modules, their sequence for the topology selection and for testing were programed into MATLAB®. The topology selection progression was already described, now the way of testing it must be discriminated. A data set was created (test set), then it was applied to the main program, to see if worked as expected. Data from this set was then compared to that in the library, to ascertain the correctness of the assumptions that originated the topology selector.

5.1 Test Set

A test set was created, to show there is a relation between the results of the optimizations in the M objectives and the overall performance that can be reached when using only a subset of these goals. This can prove the performance prediction assumption underlying this work's model and serve to demonstrate that the selecting mechanism functions correctly.

This test set had a similar setup as the library set. It was equal regarding circuits, metrics, netlists, input files, and GA parameters used. The optimizations were, however, performed for only two objectives at a time (the same as in the projections of Figure 4.6 and Figure 4.7), (1) Gain and FOM, (2) OS and VOS. These pairs were nominated due to the trade-off characteristic between some, competing with each other. Besides the optimizations also had different stopping conditions. They were arrested when there was convergence or when the population reached 6000 generations (1000 at a time). The convergence criterium was the same as the one used for the 4 objective optimizations pointed out in subsection 4.4.4. In order to increase the trustworthiness of the method, these optimizations were performed 5 times for each objective pair, guaranteeing a greater level of statistical confidence.

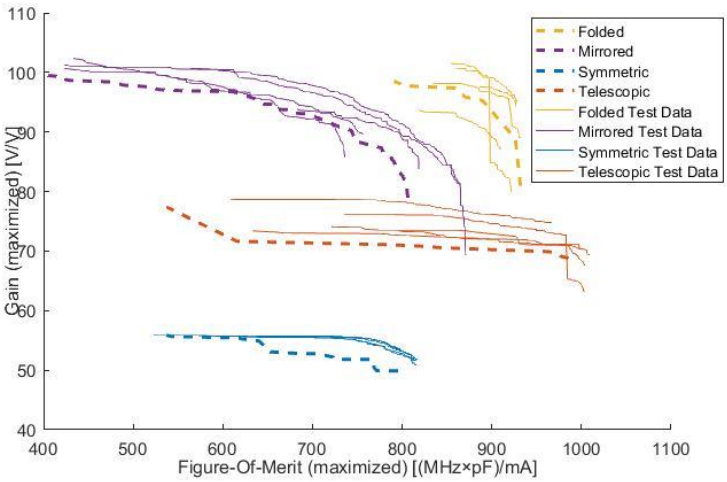

Figure 5.1 - Two objective optimization for Gain and FOM side-by-side with view of four objective pareto reformed into the same two objectives.

Due to the possibility of representing the test POFs in one single plot (2 dimensional), it becomes observable that the points are indeed formed exclusively by non-dominated points. In Figure 5.1 and Figure 5.2 the test set is represented side by side with their library equivalents. From these plots it is possible to see the curve similarities for each pair of objectives in the test set (the full lines), to the library optimizations, that were first subject to the Data Filtering and Pareto Reformation modules for the same two pairs (thicker, dotted lines). From Figure 4.6 one could see that the results were in accordance to the ones presented in Table 4.1, but Figure 4.7 was too chaotic for any elations. However, in Figure 5.2 after pareto reformation it is also visible that the OS and VOS results are as expected. This seems to point towards the utility of the modules used.

Because of some data overlap in Figure 5.2 another plot is given with Figure 5.3 that shows the library reformed pareto for the OS and VOS pair.

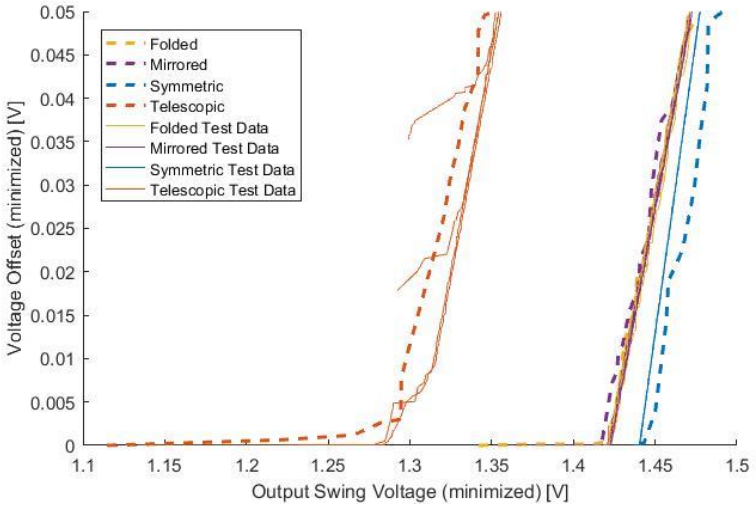

Figure 5.2 - Two objective optimization for OS and VOS side-by-side with view of four objective pareto reformed into the same two objectives.

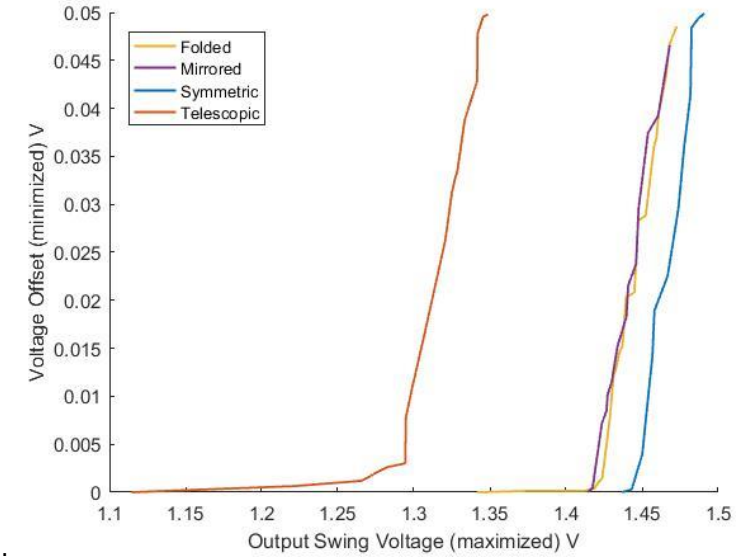

Figure 5.3 - 4 objective library reforned pareto for OS and VOS objectives.

5.2 Model validation

To prove the idea of relationship between optimizations referred in the throughout this work, a quantitative analysis was performed. In this analysis it is found a modified version of topology selection tool. This slight modification distinguishes the testing program from the original, by always employing a pure distance metric, after the pareto was reformed, and always considering the entirety of solutions. This contrasts with the original, that in the no solution occasion does not take into account coordinates that reach the goal, and which only considers goal achieving simulations if any exist. Essentially this program sums up the library to paretos, adds the line segments' closest points, and computes and sorts the distances.

A MATLAB® script was developed to iterate over all optimizations, and the simulations' output in them, entering them as input to the test program. The outcome is registered and processed to incorporate the statistics that will serve as scale of the similarity amongst the four and two objective optimizations. Different possibilities to utilize these outcomes were explored, which will now be reported.

5.2.1 Closest topology

This approach relies exclusively on what was found to be the nearest library topology to the test individual by using as input (desired specifications), in the test program. The input closest library pareto is the one more closely related to this point. Therefore, if the selected topology matches that of the analyzed instance, then the similarity is verified. The larger the number of points that the topology pairs with the one in the library, the more closely related the curves are. Hence, for each topology the success is measured through

the rate of topology matching, with test population size pop_{kij}, and the number of individuals that matched hit_{kij}, where i is the optimization number and j the topology and k the objective pair:

$$match\ rate_{kij} = \frac{hit_{kij}}{pop_{kij}} \tag{5.1}$$

The rate for an objective pair for a given topology, considering the 5 optimizations made is:

$$match\ rate_{kj} = \frac{\sum_{i=1}^{5} hit_{kij}}{\sum_{i=1}^{5} pop_{kij}} \tag{5.2}$$

For an objective pair:

$$match\ rate_{k} = \frac{\sum_{j=1}^{4}\sum_{i=1}^{5} hit_{kij}}{\sum_{j=1}^{4}\sum_{i=1}^{5} pop_{kij}} \tag{5.3}$$

And to get an overall rate:

$$total\ match\ rate = \frac{\sum_{k=1}^{2}\sum_{j=1}^{4}\sum_{i=1}^{5} hit_{kij}}{\sum_{k=1}^{2}\sum_{j=1}^{4}\sum_{i=1}^{5} pop_{kij}} \tag{5.4}$$

With:

$$0 < k \leq 2, 1 \leq j \leq 4, 0 \leq i \leq 5$$

The application of (5.4), led to:

$$total\ match\ rate = 76.46\ \%$$

The rate by objective pairs using (5.3) is displayed on Figure 5.4.

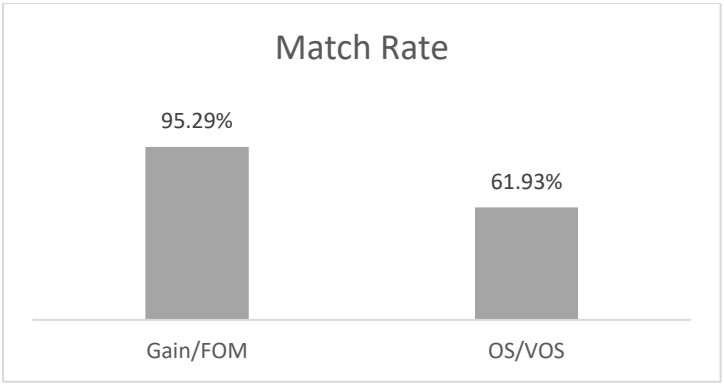

Figure 5.4 – Match Rate on the two objective pairs available (1) Gain/Fom and, (2) OS, VOS.

Portraying the significant difference in between the Gain/FOM objective, with much higher proportion of test points that are closer to the same topology in the library, than OS/VOS optimizations. Each of these pairs is further inspected by performing (5.2) whose output can be seen in Figure 5.5.

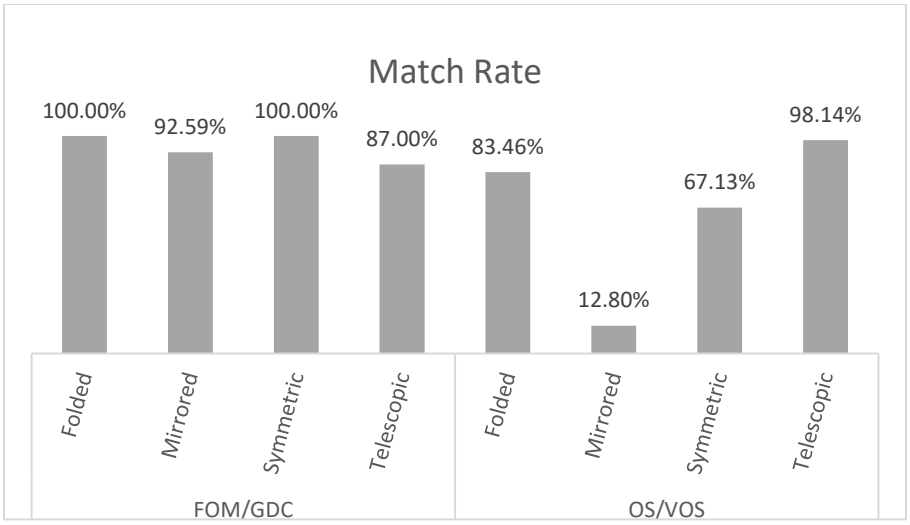

Figure 5.5 - Matching percentage in both objective pairs, per topology.

Where the rate of matching in the Mirrored is extremely low (12.80%), with the Symmetric being low as well (67.13%), both below the $total\ matching\ rate$ (76.46%). By regarding Figure 5.3 it can be noticed that the Folded and Mirrored curves are partially overlapping and are similar for the rest of the values.. In Figure 5.2 the test curves of the same two topologies and the 4 objective reduced Folded (visible on Figure 5.3), all coincide, in such a way that they are indistinguishable. The Mirrored can still be recognized in a good portion of the graphs, thus explaining the low rates. The Symmetrical test data is in between the Symmetrical and Folded paretos, sometimes closer to the latter one, also affecting the matching rate. In these same figures it is observable that the Telescopic circuit performed quite differently, from the rest of the alternatives. In Table 5.1 it is possible to see the results obtained in all optimizations, detailing the population sizes, and how many were correctly guessed (hit). These numbers were the basis for reaching the aggregate numbers seen before.

5.2.2 Average Distance

Due to the observations of performance discrepancy stated in the end of the previous subsection, there was an indication that solely giving the closest topology was insufficient. Therefore, the average distance to all library paretos was calculated, taking the closest point from each to the goal. Since all distances are positive, the average distance is only the mean of the distances for a pair of objectives in a topology. Using the normalized distances to topologies in ordered solution set S from subsection 0, $solutionsD_{tp}$ where tp is the test point individual from pop_{kij} simulations, for the ith optimization relative to objectives k in topology j, it can be extracted:

$$Average\ distances_{kj} = \frac{\sum_{i=1}^{5}\sum_{tp=1}^{pop_{kij}} solutionsD_{tp}}{\sum_{i=1}^{5} pop_{kij}} \tag{5.5}$$

Table 5.1 – Prediction break down by optimization.

		Objective Pair(k)					
		Gain/FOM			VOS/OS		
Topology(j)	Optimization(i)	# hit	Pop	HitRate	# hit	Pop	HitRate
Folded	1	66	66	100.00%	108	127	85.04%
	2	41	41	100.00%	99	108	91.67%
	3	37	37	100.00%	77	84	91.67%
	4	32	32	100.00%	63	89	70.79%
	5	71	71	100.00%	92	118	77.97%
	Total	247	247	100.00%	439	526	83.46%
Mirrored	1	112	112	100.00%	7	109	6.42%
	2	99	128	77.34%	7	125	5.60%
	3	115	115	100.00%	39	127	30.71%
	4	127	127	100.00%	22	128	17.19%
	5	109	125	87.20%	4	128	3.13%
	Total	562	607	92.59%	79	617	12.80%
Symmetric	1	127	127	100.00%	81	128	63.28%
	2	121	121	100.00%	89	130	68.46%
	3	96	96	100.00%	89	133	66.92%
	4	125	125	100.00%	91	132	68.94%
	5	118	118	100.00%	87	128	67.97%
	Total	587	587	100.00%	437	651	67.13%
Telescopic	1	42	42	100.00%	127	127	100.00%
	2	41	41	100.00%	0	8	0.00%
	3	73	73	100.00%	130	130	100.00%
	4	56	92	60.87%	129	129	100.00%
	5	29	29	100.00%	37	37	100.00%
	Total	241	277	87.00%	423	431	98.14%

Since the set contains distances to all library topologies, the average distances will contain the average distance from a k performed optimization to every pareto in the library, as seen in Figure 5.6,Figure 5.7:

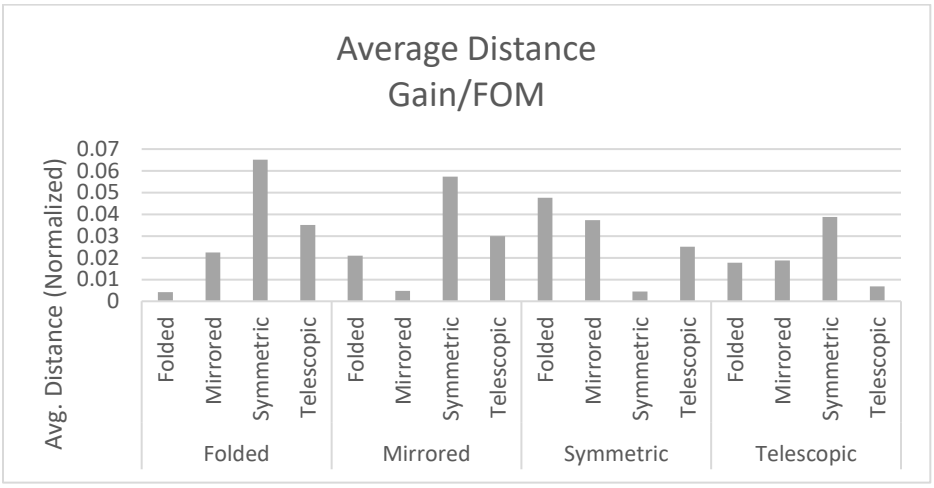

Figure 5.6 - Average distance grouped by topology of test points to library topologies for Gain/FOM objective pair.

This analysis provides valuable information, seen on Figure 5.7, where it is noticeable that the Folded, Mirrored and Symmetric topologies are much more similar than they are to the Telescopic one. Such a gap is evident from inspecting the order of magnitude of the distances, hence leading to the inclusion of the normalized distances in the output of the program. In the case of the Folded, Mirrored and Symmetric topologies, if further optimizing the first ranked topology is not sufficient, attempting the other two is much more likely to result in a wanted outcome than trying the Telescopic circuit, as evidenced by the relative distances. In Figure 5.6 these distances mirror the more balanced intervals between the paretos, making these differences, and subsequent choices, less obvious, emphasizing the first ranked element.

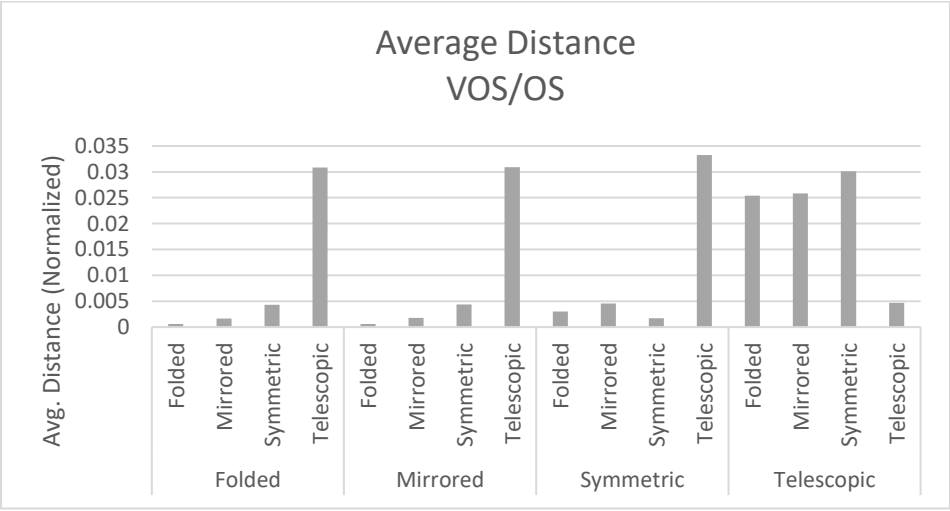

Figure 5.7 - Average distance grouped by topology of test points to library topologies for VOS/OS objective pair.

6 Conclusions

6.1 Conclusion

The AMS circuit industry faces growing challenges to keep up with the rising demands and challenges (mostly time-to-market pressures). To expedite product lifecycles, EDA tools will be increasingly resorted to. While there are some projects that tackle different phases of the design flow, when it comes to the component of topology creation the state-of-art is lacking, making it a crucial subject to undertake. In this work a new approach was developed to contribute into this field.

The following objectives, were stated in the introductory section, and had to be completed to consider this endeavor successful, were reached:

- The most significant approaches to topology creation were overviewed.
- A methodology for selection of best topologies for input goals, in both the cases of having and lacking feasible solutions was developed.
- A library of circuits and measures that could be used to test the tool's functionality was defined putting all its topologies through optimizations.
- A test set to investigate the tool's efficacy was created and applied to it.
- AIDA-TOP tool implemented.
- The results from using the test set on the tool were scrutinized.

Upon the conclusion of all the objectives it is possible to declare that the work was successful, delivering results that allow moderate optimism in considering the possibility of using the methods described as an electronic circuit assistant. The classification of AIDA-TOP as it was done for previous works in Table 2.2 is present in Table 6.1. Although the setup required is not the most appealing, this solution could use previously optimized circuits and integrated into the common optimization tasks for more seamless database creation.

Table 6.1 – AIDA-TOP Classification

Discovery	**Main Algorithm**	**Topology View**	**Evaluation**	**Objectives**	**Design Stages**	**Number of Topologies**	**Setup /Run Time**
Selection	Automatic Knowledge -based	Flat	Simulation	Multi-Objective	Independent, Cell	4	2 days/few seconds

This initial investigation into such tool requires further testing and adaptations to its in-market use. In the next subsection some possible complements to this work are suggested.

6.2 Future work

This work introduces a method for topology selection that departs significantly from previous ones, and the tests that were done to it are also the initial trials into its accuracy. Thus, it would be beneficial to further introduce topologies and metrics that could further assess the capabilities and pitfalls of the tool's current form. When it comes to the measurements, commonly used and important ones like Area and Noise would be important to test for. As for the circuits, to widen the library to consider families of circuits such as, OpAmps, low-noise amplifiers and comparators could help confirm the tool's capacity to incorporate a wide variety of topologies.

This tool could also be added to the existing family of AIDA software, placing it as an optional feature before proceeding into sizing circuits. Furthermore, this tool could be integrated in a way that would take advantage of all the optimizations for which AIDA is used for, automatically storing everything required into the library. This could facilitate the acquisition of great amounts of information to not only create a complete library, but to also use this information to improve the selection tool. To prevent the tool from storing redundant circuits, an isomorphism algorithm could be added, such as the one seen in FEATS [44], which as the name indicates detects equal circuits or equivalent circuits.

Given that the pareto reformation module always stores the same information whenever a topology is deemed as non-feasible, which is represented by the index of the corresponding non-dominated solutions, then saving this information could save on the tool's computation time for a lot of cases, while occupying a small amount of memory.

Bibliography

[1] "WSTS," *World Semiconductor Trade Statistics (WTSS) Historical Billings Report*. [Online]. Available: https://www.wsts.org/esraCMS/extension/media/f/BBH/3959/bbhist-33.xls. [Accessed: 20-Apr-2019].

[2] "Gartner Market 2018," *Gartner Says Worldwide Semiconductor Revenue Grew 13.4 Percent in 2018; Increase Driven by Memory Market*. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2019-01-07-gartner-says-worldwide-semiconductor-revenue-grew-13-. [Accessed: 20-Apr-2019].

[3] "Qualcomm Datacenter Technologies Announces Commercial Shipment of Qualcomm Centriq 2400," *Qualcomm*, 08-Nov-2017. [Online]. Available: https://www.qualcomm.com/news/releases/2017/11/08/qualcomm-datacenter-technologies-announces-commercial-shipment-qualcomm. [Accessed: 28-Apr-2019].

[4] L. Lavagno, I. L. Markov, G. Martin, and L. K. Scheffer, Eds., "Section I: Introduction," in *Electronic Design Automation for Ic System Design, Verification, and Testing*, 2 edition., Boca Raton: CRC Press, 2016.

[5] L. Lavagno, I. L. Markov, G. Martin, and L. K. Scheffer, Eds., "Chapter 1: Design Flows," in *Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology*, 2 edition., Boca Raton: CRC Press, 2016.

[6] L. Lavagno, I. L. Markov, G. Martin, and L. K. Scheffer, Eds., "Chapter 19: Layout Tools for Analog Integrated Circuits and Mixed-Signal Systems-on-Chip," in *Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology*, 2 edition., Boca Raton: CRC Press, 2016.

[7] "McClean Report Contents and Summaries 2019 | IC Insights." [Online]. Available: http://www.icinsights.com/services/mcclean-report/report-contents/. [Accessed: 27-Apr-2019].

[8] N. Lourenço, R. Martins, and N. C. G. Horta, *Automatic Analog IC Sizing and Optimization Constrained with PVT Corners and Layout Effects*. Springer International Publishing, 2017.

[9] G. G. E. Gielen and R. A. Rutenbar, "Computer-aided design of analog and mixed-signal integrated circuits," *Proc. IEEE*, vol. 88, no. 12, pp. 1825–1854, Dec. 2000, doi: 10.1109/5.899053.

[10] G. E. Moore, "The role of Fairchild in silicon technology in the early days of 'Silicon Valley,'" *Proc. IEEE*, vol. 86, no. 1, pp. 53–62, Jan. 1998, doi: 10.1109/5.658759.

[11] R. Martins, N. Lourenço, and N. Horta, "LAYGEN II—Automatic Layout Generation of Analog Integrated Circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 32, no. 11, pp. 1641–1654, Nov. 2013, doi: 10.1109/TCAD.2013.2269050.

[12] R. Lourenço, N. Lourenço, and N. Horta, *AIDA-CMK: Multi-Algorithm Optimization Kernel Applied to Analog IC Sizing*. Springer International Publishing, 2015.

[13] "EDA Tools and IP for Intelligent System Design | Cadence." [Online]. Available: https://www.cadence.com/content/cadence-www/global/en_US/home.html. [Accessed: 27-Sep-2019].

[14] "Synopsys." [Online]. Available: https://www.synopsys.com/. [Accessed: 27-Sep-2019].

[15] N. Lourenço, R. Martins, A. Canelas, R. Póvoa, and N. Horta, "AIDA: Layout-Aware Analog Circuit-Level Sizing with In-Loop Layout Generation," *Integr. VLSI J.*, vol. 55, May 2016, doi: 10.1016/j.vlsi.2016.04.009.

[16] T. McConaghy, P. Palmers, M. Steyaert, and G. G. E. Gielen, "Trustworthy Genetic Programming-Based Synthesis of Analog Circuit Topologies Using Hierarchical Domain-Specific Building Blocks," *IEEE Trans. Evol. Comput.*, vol. 15, no. 4, pp. 557–570, Aug. 2011, doi: 10.1109/TEVC.2010.2093581.

[17] T. Sripramong and C. Toumazou, "The invention of CMOS amplifiers using genetic programming and current-flow analysis," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 21, no. 11, pp. 1237–1252, Nov. 2002, doi: 10.1109/TCAD.2002.804109.

[18] T. R. Dastidar, P. P. Chakrabarti, and P. Ray, "A synthesis system for analog circuits based on evolutionary search and topological reuse," *IEEE Trans. Evol. Comput.*, vol. 9, no. 2, pp. 211–224, Apr. 2005, doi: 10.1109/TEVC.2004.841308.

[19] M. G. R. Degrauwe *et al.*, "IDAC: an interactive design tool for analog CMOS circuits," *IEEE J. Solid-State Circuits*, vol. 22, no. 6, pp. 1106–1116, Dec. 1987, doi: 10.1109/JSSC.1987.1052861.

[20] J. Rijmenants, J. B. Litsios, T. R. Schwarz, and M. G. R. Degrauwe, "ILAC: an automated layout tool for analog CMOS circuits," *IEEE J. Solid-State Circuits*, vol. 24, no. 2, pp. 417–425, Apr. 1989, doi: 10.1109/4.18603.

[21] R. Harjani, R. A. Rutenbar, and L. R. Carley, “OASYS: A Framework for Analog Circuit Synthesis,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 8, no. 12, pp. 1247–1266, Jan. 1989, doi: 10.1109/43.44506.
[22] H. Y. Koh, C. H. Sequin, and P. R. Gray, “OPASYN: a compiler for CMOS operational amplifiers,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 9, no. 2, pp. 113–125, Feb. 1990, doi: 10.1109/43.46777.
[23] P. C. Maulik, L. R. Carley, and R. A. Rutenbar, “A mixed-integer nonlinear programming approach to analog circuit synthesis,” in *[1992] Proceedings 29th ACM/IEEE Design Automation Conference*, 1992, pp. 698–703, doi: 10.1109/DAC.1992.227796.
[24] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear programming: theory and algorithms*, 3. ed. Hoboken, NJ: Wiley-Interscience, 2006.
[25] J. D. Lohn and S. P. Colombano, “A circuit representation technique for automated circuit design,” *IEEE Trans. Evol. Comput.*, vol. 3, no. 3, pp. 205–219, Sep. 1999, doi: 10.1109/4235.788491.
[26] D. Ashlock, “Chapter 1,” in *Evolutionary Computation for Modeling and Optimization*, New York: Springer-Verlag, 2006.
[27] J. R. Koza, “Chapter 3: Introduction to Genetic Algorithms,” in *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, 1 edition., Cambridge, Mass: A Bradford Book, 1992.
[28] J. R. Koza, “Chapter 4: Representation Problem for Genetic Algorithms,” in *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, 1 edition., Cambridge, Mass: A Bradford Book, 1992.
[29] K. Deb and D. Kalyanmoy, “Chapter 2: Multi-Objective Optimization,” in *Multi-Objective Optimization Using Evolutionary Algorithms*, New York, NY, USA: John Wiley & Sons, Inc., 2001.
[30] K. Deb and D. Kalyanmoy, “Chapter 4: Evolutionary Algorithms,” in *Multi-Objective Optimization Using Evolutionary Algorithms*, New York, NY, USA: John Wiley & Sons, Inc., 2001.
[31] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002, doi: 10.1109/4235.996017.
[32] D. Vernon, *Artificial Cognitive Systems: A Primer*. MIT Press, 2014.
[33] H. Li, X. Liu, F. Jiao, A. Doboli, and S. Doboli, “InnovA: A Cognitive Architecture for Computational Innovation Through Robust Divergence and Its Application for Analog Circuit Design,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 10, pp. 1943–1956, Oct. 2018, doi: 10.1109/TCAD.2017.2783344.
[34] F. El-Turky and E. E. Perry, “BLADES: an artificial intelligence approach to analog circuit design,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 8, no. 6, pp. 680–692, Jun. 1989, doi: 10.1109/43.31523.
[35] B. J. Sheu, J. C. Lee, and A. H. Fung, “Flexible architecture approach to knowledge-based analogue IC design,” *IEE Proc. G - Circuits Devices Syst.*, vol. 137, no. 4, pp. 266–274, Aug. 1990, doi: 10.1049/ip-g-2.1990.0041.
[36] Z.- Ning, T. Mouthaan, and H. Wallinga, “SEAS: a simulated evolution approach for analog circuit synthesis,” in *Proceedings of the IEEE 1991 Custom Integrated Circuits Conference*, 1991, pp. 5.2–1, doi: 10.1109/CICC.1991.164025.
[37] H. Chang *et al.*, “A Top-down, Constraint-driven Design Methodology For Analog Integrated Circuits,” in *1992 Proceedings of the IEEE Custom Integrated Circuits Conference*, 1992, pp. 8.4.1-8.4.6, doi: 10.1109/CICC.1992.591162.
[38] J. B. Grimbleby, “Automatic analogue network synthesis using genetic algorithms,” in *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, 1995, pp. 53–58, doi: 10.1049/cp:19951024.
[39] D. L. W. Kruiskamp, “DARWIN: CMOS opamp Synthesis by Means of a Genetic Algorithm,” in *32nd Design Automation Conference*, 1995, pp. 433–438, doi: 10.1145/217474.217566.
[40] A. J. Torralba, J. Chavez, and L. G. Franquelo, “Fuzzy-logic-based analog design tools,” *IEEE Micro*, vol. 16, no. 4, pp. 60–68, Aug. 1996, doi: 10.1109/40.526926.
[41] J. R. Koza, F. H. Bennett, D. Andre, M. A. Keane, and F. Dunlap, “Automated synthesis of analog electrical circuits by means of genetic programming,” *IEEE Trans. Evol. Comput.*, vol. 1, no. 2, pp. 109–128, Jul. 1997, doi: 10.1109/4235.687879.

[42] I. O'Connor and A. Kaiser, "Automated design of switched-current cells," in *Proceedings of the IEEE 1998 Custom Integrated Circuits Conference (Cat. No.98CH36143)*, 1998, pp. 477–480, doi: 10.1109/CICC.1998.695022.
[43] G. V. der Plas *et al.*, "AMGIE-A synthesis environment for CMOS analog integrated circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 9, pp. 1037–1058, Sep. 2001, doi: 10.1109/43.945301.
[44] M. Meissner and L. Hedrich, "FEATS: Framework for Explorative Analog Topology Synthesis," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 2, pp. 213–226, Feb. 2015, doi: 10.1109/TCAD.2014.2376987.
[45] A. Gerlach, J. Scheible, T. Rosahl, and F.-T. Eitrich, "A generic topology selection method for analog circuits with embedded circuit sizing demonstrated on the OTA example," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, Lausanne, Switzerland, 2017, pp. 898–901, doi: 10.23919/DATE.2017.7927115.
[46] C.-L. Hwang and K. Yoon, "Section III. 2.3.5. TOPSIS," in *Multiple Attribute Decision Making: Methods and Applications A State-of-the-Art Survey*, Berlin, Heidelberg: Springer Berlin Heidelberg, 1981.
[47] A. S. Sedra and K. C. Smith, "Chapter 5: Operational Amplifiers," in *Microelectronic circuits: theory and applications*, Oxford University Press, 2009.
[48] R. F. S. Póvoa, J. C. da P. Goes, and N. C. G. Horta, *A New Family of CMOS Cascode-Free Amplifiers with High Energy-Efficiency and Improved Gain*. Springer International Publishing, 2019.
[49] T. C. Carusone, D. Johns, and K. Martin, *Analog Integrated Circuit Design*, 2 edition. Hoboken, NJ: Wiley, 2011.
[50] K. Deb and D. Kalyanmoy, "Chapter 1: Prologue," in *Multi-Objective Optimization Using Evolutionary Algorithms*, New York, NY, USA: John Wiley & Sons, Inc., 2001.
[51] A. Canelas, R. Martins, R. Póvoa, N. Lourenço, and N. Horta, "Yield optimization using k-means clustering algorithm to reduce Monte Carlo simulations," in *2016 13th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2016, pp. 1–4, doi: 10.1109/SMACD.2016.7520729.
[52] R. Povoa, N. Lourenco, R. Martins, A. Canelas, N. C. G. Horta, and J. Goes, "Single-Stage Amplifier Biased by Voltage Combiners With Gain and Energy-Efficiency Enhancement," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 65, no. 3, pp. 266–270, Mar. 2018, doi: 10.1109/TCSII.2017.2686586.
[53] R. Povoa, N. Lourenco, R. Martins, A. Canelas, N. Horta, and J. Goes, "Single-Stage OTA Biased by Voltage-Combiners With Enhanced Performance Using Current Starving," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 65, no. 11, pp. 1599–1603, Nov. 2018, doi: 10.1109/TCSII.2017.2777533.
[54] R. Povoa, N. Lourenco, R. Martins, A. Canelas, N. Horta, and J. Goes, "A Folded Voltage-Combiners Biased Amplifier for Low Voltage and High Energy-Efficiency Applications," *IEEE Trans. Circuits Syst. II Express Briefs*, pp. 1–1, 2019, doi: 10.1109/TCSII.2019.2913083.