SPRINGER BRIEFS IN APPLIED SCIENCES AND TECHNOLOGY · COMPUTATIONAL INTELLIGENCE

Ricardo Lourenço
Nuno Lourenço
Nuno Horta

AIDA-CMK: Multi-Algorithm Optimization Kernel Applied to Analog IC Sizing

Springer

SpringerBriefs in Applied Sciences and Technology

Computational Intelligence

Series editor

Janusz Kacprzyk, Warsaw, Poland

About this Series

The series "Studies in Computational Intelligence" (SCI) publishes new developments and advances in the various areas of computational intelligence—quickly and with a high quality. The intent is to cover the theory, applications, and design methods of computational intelligence, as embedded in the fields of engineering, computer science, physics and life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in computational intelligence spanning the areas of neural networks, connectionist systems, genetic algorithms, evolutionary computation, artificial intelligence, cellular automata, self-organizing systems, soft computing, fuzzy systems, and hybrid intelligent systems. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution, which enable both wide and rapid dissemination of research output.

More information about this series at http://www.springer.com/series/10618

Ricardo Lourenço · Nuno Lourenço
Nuno Horta

AIDA-CMK: Multi-Algorithm Optimization Kernel Applied to Analog IC Sizing

Springer

Ricardo Lourenço
Instituto de Telecomunicações, Instituto Superior Técnico
Lisbon
Portugal

Nuno Lourenço
Instituto de Telecomunicações, Instituto Superior Técnico
Lisbon
Portugal

Nuno Horta
Instituto de Telecomunicações, Instituto Superior Técnico
Lisbon
Portugal

ISSN 2191-530X ISSN 2191-5318 (electronic)
SpringerBriefs in Applied Sciences and Technology
ISBN 978-3-319-15954-6 ISBN 978-3-319-15955-3 (eBook)
DOI 10.1007/978-3-319-15955-3

Library of Congress Control Number: 2015932844

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

To Sílvia

Ricardo Lourenço

To Alina

Nuno Lourenço

To Carla, João and Tiago

Nuno Horta

Preface

The rapid evolution and widespread use of consumer electronic devices such as cell phones, tablets, or smart TV puts a great innovative pressure on the integrated circuit industry. Most of the functionalities of such devices are implemented using digital circuitry but analog tasks such as converting, receiving, and emitting signals, regulating power or communicating to device sensors still play a major role in modern ICs, leading to mixed-signal systems. However, the design automation of analog circuits is far behind that of the digital circuits, either in techniques or in tools, failing to help designers to compete with the demanding time-to-market, lowering the costs and cutting development time. In this context, AIDA Framework fully developed at the Integrated Circuits Group from Instituto de Telecomunicações, Lisbon, Portugal, appears as an Electronic Design Automation tool to aid designers to do their job better and faster. This work focuses on AIDA-CMK, by enhancing AIDA-C, which is the circuit optimizer component of AIDA, with a new multi-objective multi-constraint optimization module that constructs a base for multiple algorithm implementations. In the proposed solution, three approaches to multi-objective multi-constraint optimization, namely, an evolutionary approach with NSGA-II, a swarm intelligence approach with MOPSO, and stochastic hill climbing approach with MOSA are implemented. Moreover, the implemented structure allows the easy hybridization between kernels transforming the previous simple NSGA-II optimization module into a more evolved and versatile module supporting multiple single and multi-kernel algorithms. The three multi-objective optimization approaches were validated with CEC2009 benchmarks to constrained multi-objective optimization and tested with real analog IC design problems. The achieved results were compared in terms of performance, using statistical results obtained from multiple independent runs showing that NSGA-II outperforms the other two single kernel reference approaches by having a better convergence time, a widespread set of solutions, and, in general, achieving better Pareto fronts. Finally, some hybrid approaches were also experimented, giving a foretaste to a wide range of opportunities to explore in future work.

The book is organized in seven chapters.

Chapter 1 presents a brief introduction to the area of analog IC design automation, with special emphasis to the automatic circuit sizing. First, the analog design problem is characterized, then, a well-accepted design flow for analog IC is presented, and finally, AIDA-CMK features are outlined.

Chapter 2 presents an overview of the state of the art in analog circuit optimization, focusing on the optimization approaches that are used. This study is then used to select the optimization methods to be considered in the framework to be developed.

Chapter 3 presents AIDA-CMK and the circuit optimization, describing the proposed architecture for the multi-objective circuit optimization framework. Also, the NSGA-II, MOSA, and MOPSO algorithms implemented are described.

Chapter 4 details the implementation, showing the application layers and their implementation. The structure of the classes implemented is described in detail showing their relations and the flexibility of the implemented framework.

Chapter 5 shows the results obtained from applying the implemented algorithms to the constrained problems from CEC 2009 competition. These analytical benchmarks are used to tune the algorithms' parameters and make a preliminary assessment of their performances.

Chapter 6 applies the developed algorithms to the optimization of real analog IC designs, showing that the behavior of the algorithms strongly differs from that that was obtained using the analytical benchmarks.

Chapter 7 addresses the conclusions and some directions for future developments are suggested.

Ricardo Lourenço
Nuno Lourenço
Nuno Horta

Contents

Abbreviations

AMS	Analog and Mixed-Signal
CAD	Computer-Aided Design
CMOS	Complementary Metal Oxide Semiconductor
DSP	Digital Signal Processing
EDA	Electronic Design Automation
GA	Genetic Algorithm
GUI	Graphical User Interface
IC	Integrated Circuit
MOEA	Multi-objective Evolutionary Algorithm
POF	Pareto Optimal Front
PSO	Particle Swarm Optimization
SA	Simulated Annealing
SoC	System-on-a-Chip
VLSI	Very-large-scale Integration

Chapter 1
Introduction

Abstract This chapter introduces analog integrated circuits (ICs) design and design automation. The chapter starts by framing analog ICs in the semiconductors' industry and a brief introduction on how they are designed. Then, the motivation for research in their design automation, and finally, the research goals of this work that focus on analog IC sizing optimization are outlined.

Keywords Analog IC design · Automatic circuit sizing · Circuit optimization · Electronic design automation · Computer-aided design

1.1 Analog IC Design

Very-large-scale integration (VLSI) technologies keep improving, enabling the steady growth of integrated circuit (IC) market. The increasing need of faster, optimized, and reliable electronic devices urges the cost-effective development of such devices to efficiently meet customers' demand under highly competitive time-to-market pressure.

Today's electronic systems are extremely complex multimillion transistor ICs. This complexity is only possible because the designers are assisted by computer-aided design (CAD) tools that support the design process.

Because analog ICs are very sensitive, they are difficult to design and reuse, consequently designers have been replacing functions of analog circuits for digital computing whenever possible. While most of the functions are implemented using digital or digital signal processing (DSP) units, some functions remain analog, and are integrated together with the digital part leading to mixed-signal systems-on-chip (SoC) designs. Some of the functionalities that will remain analog are listed below.

- *Sensing the systems inputs*: the signals of a sensor, microphone, or antenna have to be detected or received, amplified, and filtered, to enable digitalization with good signal-to-noise and distortion ratio. Typical applications of these circuits are in sensor interfaces, telecommunication receivers, or sound recording;

R. Lourenço et al., *AIDA-CMK: Multi-Algorithm Optimization Kernel Applied to Analog IC Sizing*, SpringerBriefs in Computational Intelligence,
DOI 10.1007/978-3-319-15955-3_1

- *Converting analog signals to digital signals*: mixed-signal circuits such as sample-and-hold, analog-to-digital converters, phase-locked loops, and frequency synthesizers provide the interface between the input/output of a system and digital processing parts of a SoC;
- *Converting the digital output back to analog*: the signal from digital processing must be converted and strengthened to analog so the signal can be conducted to the output with low distortion;
- *Provide and regulate power*: voltage/current reference circuits and crystal oscillators offer stable and absolute references for the sample-and-hold, analog-to-digital converters, phase-locked loops, and frequency synthesizers;
- *The implementation at transistor level of the digital gates*: the last kind of analog circuits are extremely high-performance digital circuits. As exemplified by the microprocessors, custom sized as analog circuits, for achieving the highest speed and the lowest power consumption.

Telecommunications, medical, and multimedia applications make extensive use of electronic devices where blocks of analog-mixed signal (AMS), digital processors, and memory blocks are integrated together [1, 2]. The increase in the performance of ICs is mostly supported by an exponential increase in the density of transistors present in ICs while inversely reducing the transistors' cost, as described by Moore's law [3]. Moore's law states that every 2 years the density of transistors on ICs doubles at the same cost. Moore's law is a law of economics not physics, and Moore itself already preconized its end, as such an exponential law "can't continue forever," but it is still valid today.

Despite the advantages that the new fabrication technologies with their increased device density have to system performance, such reduction of device sizes decrease their intrinsic analog performance, while increasing variability and imposing extra constraints to the circuits design, further impairing their productivity.

As stated before the analog parts in a SoC occupy only approximately 20 % of the global circuit area (as shown in Fig. 1.1) but the design effort is considerably higher in comparison to the design effort of the digital section. In the digital IC design, several mature electronic design automation (EDA) tools and design methodologies are available that help the designers keeping up with the new capabilities offered by the technology, and, making circuit reuse usual, leading to an increased design productivity. By contrast and despite the algorithms and techniques introduced in the last 25 years, in analog design there are no mature and well-defined strategies to address a problem, leading to custom solutions that are difficult to reuse.

Given the giant growth of AMS systems, pressed by the need of electronic products, which are affordable and reliable, and developed under very strict time-to-market constraints, the development and improvement of CAD tools that increase analog designers' productivity and the quality of the resulting designs is an urgent need.

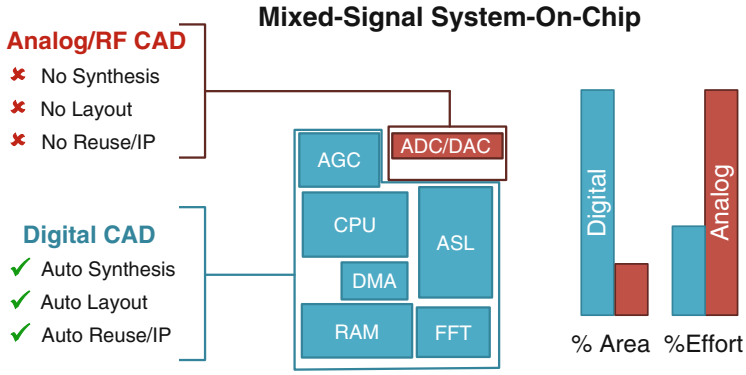

Fig. 1.1 Digital versus analog design reality [4]

1.2 The Analog IC Design Flow

One of the problems with analog design automation is that the exact design flow varies among designers, projects, and companies. Nevertheless, a design flow well known and generally accepted for analog-mixed signal ICs is described in Fig. 1.2. This design flow was introduced by Gielen and Rutenbar [2], which consists of a series of top-down design selection and specifications step by step, from system level to the device level, and bottom up layout generation and verification.

At the circuit level, a design task is performed for each analog block. This process is executed iteratively in order to determine the physical dimensions of each device. In this phase are considered two major tasks: the selection of circuit topology and the circuit sizing where the design parameters of the electronic devices are defined. The specifications are then, verified through simulation of the circuit by, e.g., HSPICE® [5] or Spectre® [6]. After the analog blocks have been sized, the project enters into the next phase where the analog blocks are mapped into a physical representation of the circuit, the layout. The layout is a set of geometric shapes that obey to the rules defined by the manufacturing process. Then, the layout passes the design rule check and the layout-versus-schematic verification stages, and finally, it is extracted and simulated to verify the impact of layout parasitic effects on the overall performance of the circuit.

Due to the lack of automation, designers keep exploring the solution space manually. This method causes long design times, and allied to the nonreusable nature of analog IC, makes analog IC design a cumbersome task. This difference in the level of automation between analog and digital design is because analog in general is less systematic, more heuristic, and knowledge intensive than the digital counterpart, and becomes critic when digital and analog circuits are integrated together. As the analog automation tools do not progress at the same pace of technology, knowledge and experience of the designer is always crucial for making decisions at all stages of the analog design flow.

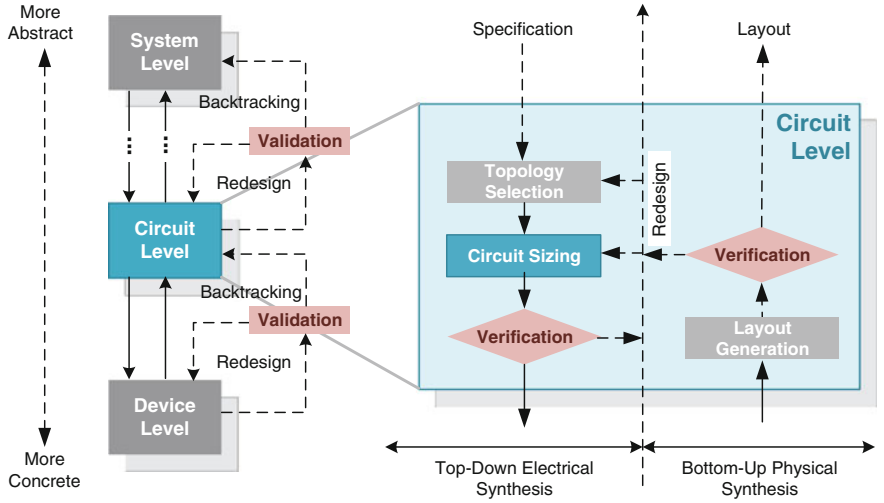

Fig. 1.2 From system level to device level tasks of analog integrated circuit design [2]

In a traditional analog design, the designer defines a methodology; interacts manually with the proper tools in order to achieve the project objectives, whether they are the best sizing of circuit parameters to meet the desired performance specifications, either to optimize the parameters for specific application (DC Gain, power, area, etc.), aiming to obtain at the end a robust design. However, the search space of the objective function, which relates the design variables to the performance specifications of the circuit, is characterized by a complex multidimensional and irregular space, making the manual search for the ideal solution difficult to achieve. Along with the time constraints faced by the designers, this search of the design space becomes very limited.

Despite its fundamental aid to designers, those tools have limited automation options, and the ones available are usually not used by the majority of the designers. The time required to manually implement an analog project is usually of weeks or months, which is in opposition to the market pressure to accelerate the release of new and high-performance ICs. To address all the difficulties to solve these problems, EDA CAD is a solution increasingly strong and solid. The main focus of this dissertation is at automatic circuit-level sizing and optimization.

1.3 Motivation

In summary, despite the trend to implement almost everything using digital circuitry, is notorious the importance of analog circuits as some parts cannot be ported to the digital world. The differences of effort and time consumption spent between digital and analog circuitry design are easy to spot, putting an enormous

pressure into the industry and academia to improve tools and methodologies to aid analog circuit designers to improve their productivity and reduce production costs. Moreover, because of the extremely large number of variations of analog circuits and fabrications processes, it is extremely difficult to make fair and accurate comparison between approaches. This is the scenario where AIDA Framework [7] appears as an EDA tool, more details about the AIDA project can be found in 'www.aidasoft.com', where AIDA-C is the tool developed for analog integrated circuit sizing and optimization.

1.4 Research Contributions

Lead by those thriving ideas, this work focus is AIDA-CMK that targets the sizing of the devices in analog circuits using state-of-the-art and innovative multi-objective optimization (MOO) techniques, by enhancing AIDA-C with a new abstraction layer that permits the easy inclusion of new multi-objective multiconstraint optimization techniques aside the NSGA-II that was originally supported. Taking advantage of the new abstraction layer, the MOO kernels, NSGA-II, MOPSO, MOSA, and the two multiple kernel hybridization are included in AIDA-CMK.

This work was supported in part by the Instituto de Telecomunicações (Research project OPERA—PEst-OE/EEI/LA0008/2013) and by the Fundação para a Ciência e Tecnologia (Research project DISRUPTIVE EXCL/EEI-ELC/0261/2012.

The specific goals of this work are:

- Implementation of a modular framework for optimizations in the scope of analog circuits optimization;
- Implement some classical MOO methods to verify the approach;
- Test the performance of different optimizations approaches:
  - Standard benchmarks;
  - Analog Integrated Circuits;
- Evaluate hybrid optimization methods.

References

1. Gielen, G.G.E.: CAD tools for embedded analogue circuits in mixed-signal integrated systems on chip. Comput. Digital Tec. IEE Proc. **152**(3), 317–332 (2005)
2. Gielen, G., Rutenbar, R.: Computer-aided design of analog and mixed-signal integrated circuits. Proc. IEEE **88**(12), 1825–1854 (2000)
3. Moore, G.E.: Cramming more components onto integrated circuits. Proc. IEEE **86**(1), 82–85 (1998)
4. Barros, M.F.M., Guilherme, J.M.C., Horta, N.C.G.: Analog circuits and systems optimization based on evolutionary computation techniques. Springer, Berlin (2010)

5. Synopsis Inc.: Synopsys HSPICE—accurate circuit simulation. http://www.synopsys.com (2014). Accessed 2014
6. Cadence: Cadence spectre accelerated parallel simulator. http://www.cadence.com (2014). Accessed 2014
7. Martins, R., Lourenço, N., Rodrigues, S., Guilherme, J., Horta, N.: AIDA: automated analog IC design flow from circuit level to layout. In: International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design, Seville, 2012

Chapter 2
Previous Works on Automated Analog IC Sizing

Abstract In the last 25 years, the scientific community proposed many techniques for the automation of analog integrated circuit sizing. In this chapter, those approaches are briefly surveyed, focusing on the optimization techniques that are used. The different approaches are classified in terms of the techniques used and the most significant aspects observed were the setup and the execution time, as well as the accuracy in the evaluation of the solutions. The study is then used to select the optimization methods to be considered in the developed framework.

Keywords Analog IC design · Optimization-based circuit sizing · Electronic design automation · Computer-aided design

2.1 Automatic IC Sizing

Analog IC sizing automation techniques are classified into two main groups, the knowledge-based approaches and the optimization-based approaches [1]. This classification is based on the fundamental techniques used to address the problem, as illustrated in Fig. 2.1.

2.1.1 Knowledge-Based Automatic Circuit Sizing

Early automation systems [2–5] did not use optimization and tried to systematize the design by using a design plan derived from expert knowledge. In these methods, a plan is built with design equations and a design strategy that produces the component sizes that meet the performance requirements. These knowledge-based approaches were applied with moderate success. The main advantage of this approach is the short execution time. However, deriving the design plan is hard and time-consuming, and the design plan requires constant maintenance in order to keep

R. Lourenço et al., *AIDA-CMK: Multi-Algorithm Optimization Kernel Applied to Analog IC Sizing*, SpringerBriefs in Computational Intelligence,
DOI 10.1007/978-3-319-15955-3_2

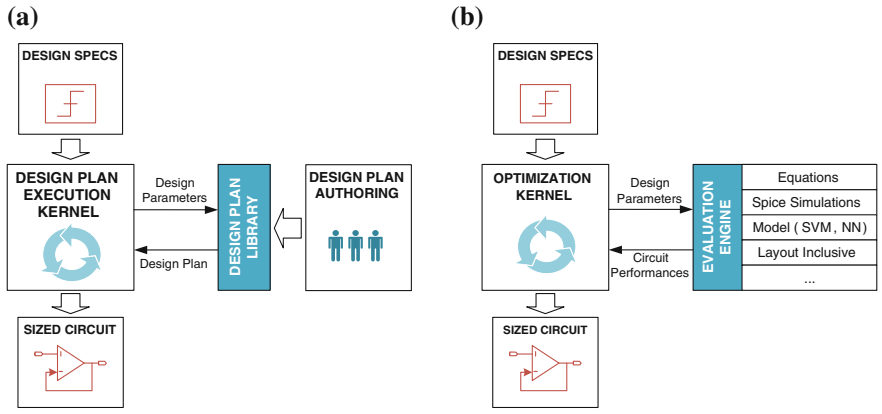

Fig. 2.1 Automatic circuit sizing approaches; **a** knowledge-based; **b** optimization-based

it up to date with technological evolution, also, the results are not optimal, suitable only as a first-cut-design.

The next generations of sizing tools apply optimization techniques to analog IC sizing. They can be further classified into two main subclasses: equation-based or simulation-based, from the method used to evaluate the circuit's performance.

2.1.2 Equation-Based Automatic Circuit Sizing

Equation-based methods use analytic expressions to relate the circuit's performance figures to the design variables. Different optimization techniques are used, both deterministic and stochastic. Knowing the equations and their properties allows the use of classical optimization methods. In OPASYN [6], the optimization is performed using steepest descent; similarly, in STAIC [7], it is used as a successive solution refinements technique.

Maulik et al. [8, 9] define the sizing problem as a constrained nonlinear optimization problem using spice models and DC operating point constraints, solving it using sequential quadratic programming. Matsukawa et al. [10] design ΔΣ and pipeline analog to digital converters solving, via convex optimization the equations that relate the performance of the converter to the size of the components.

In GPCAD [11], a posynomial circuit model is optimized using Geometrical Programming (GP); the execution time is in the order of few seconds, but the general application of posynomial models is difficult and the time to derive the model for new circuits is still high. Kuo-Hsuan et al. [12] revisited the posynomial modeling recently, surpassing the accuracy issue by introducing an additional generation step, where local optimization using simulated annealing (SA) and a circuit simulator is performed. The same strategy is applied in FASY [13, 14], where analytical

expressions are solved to generate an initial solution and a simulation-based optimization is performed to fine tune the solution.

Other equation-based approaches do not limit the problem formulation in order to use a specific optimization technique at all, relying on heuristic optimization instead. OPTIMAN [15] uses SA applied to analytical models, and, in ASTRX/OBLX [16], an SA optimization is also performed using cost function defined by equations for dc operation point, and small signal Asymptotic Waveform Evaluation (AWE)-based simulation; this evaluation technique is also used in DARWIN [17], which uses Genetic Algorithms (GA) instead. Doboli et al. [18] also apply genetic programming techniques to simultaneously derive the sub-block specifications, sub-block topology selection, and transistor sizing.

Equation-based methods' strong point is the short evaluation time, making them, like the knowledge-based approaches, extremely suited to derive first-cut designs. The main drawback is that, despite the advances in symbolic analysis, not all design characteristics can be easily captured by analytic equations, making the generalization of the method to different circuits very difficult. In addition, the approximations introduced in the equations yield low accuracy designs, especially for complex circuits, requiring additional work to ensure that the circuit really meets the specifications.

2.1.3 Simulation-Based Automatic Circuit Sizing

With the availability of computing resources, simulation-based optimization gained ground, and is the most common method found in recent approaches. In simulation-based sizing, as in the case of AIDA-C, a circuit simulator, e.g., SPICE [19], is used to evaluate the circuit performance.

Early approaches to simulation-based automatic sizing used local optimization around a "good" solution, where SA [20] is the most commonly optimization technique used. In DELIGTH.SPICE [21], the optimization algorithm (phase I-II-III method of feasible directions) is used to perform local design optimization around a user provided starting point. Kuo-Hsuan et al. [12] and FASY [13, 14] use equation-based techniques to derive an approximate solution, and then use simulation within a SA kernel to optimize the design. Likewise, Cheng et al. [22] also uses SA but considers the transistor bias conditions to constrain the problem, and, instead of solving the circuit by finding transistor sizes, the problem is solved by finding the bias of the transistors. FRIDGE [23] aims for general applicability approach by using an annealing-like optimization without any restriction to the starting point. Castro-Lopez et al. [24] use SA followed by a deterministic method for fine-tuning to perform the optimization.

Another widely used class of optimization methods is the GA. Barros et al. in [1, 25, 26] presents a circuit sizing optimization supported by a genetic algorithm where the evaluations of the populations were made using both a circuit simulator and an automatically trained support vector machine. Alpaydin et al. [27] use

hybridization of evolutionary and annealing optimization strategy where the circuits' performance figures are computed using a blend of equations and simulations.

Given the affinity evolutionary algorithms have with parallel implementations, in Santos-Tavares [28], MAELSTROM [29], and ANACONDA [30] the time to simulate the population reduced by a parallel mechanism that shares the evaluation load among multiple computers. Because the traditional use of local search methods in many implementations, the MAELSTROM's authors option was to use a hybridization, i.e., the parallel recombinative simulated annealing (PRSA). In ANACONDA the approach is similar but instead of the PRSA it is applied a variation of pattern search algorithms, named by the authors as stochastic pattern search.

A different approach to circuit sizing optimization that also employs evolutionary methods is to simultaneously generate the circuit topologies (the arrangement of the devices) and the device sizes. Koza [31], Sripramong [32], and more recently Hongying [33] proposed a design methodology able to create new topologies by exploring the immense possibilities starting from low abstraction level. Small elementary blocks are connected bottom-up to each other to form a new topology. Various fundamental entities can be applied, such as, single transistors, elementary building blocks, or node connections. However, these approaches are met with great skepticism, as designers are suspicious of the generated structures, because they often differ "too" much from the well-known analog circuit structures.

Swarm intelligence algorithms [34] can also be found in the literature applied to analog circuit sizing. The fundament of swarm intelligence algorithms is to use many simple agents that lead an intelligent global behavior, like the one observed in many insect hives. From these methods, the most commons are the ant colony optimization (ACO), which was successfully applied in [35, 36], and particle swarm optimization (PSO) that can be found in [37–39].

Circuit sizing is in its essence a multi-objective multi-constraint problem, and the designer often explores the tradeoff among contradictory performance measures, for example, minimizing power consumption while maximizing bandwidth, or maximizing gain and minimizing area of an amplifier, as such, the usage of multi-objective optimization techniques is becoming more common. When considering multiple objectives the output is not one solution, but a set of optimal design tradeoff solutions, usually referred as Pareto optimal front (POF). Given the multiple elements already present in both evolutionary and swarm intelligence algorithms, these are the natural candidates to implement such approach. In GENOM-POF [40, 41] and MOJITO [42], the evolutionary multi-objective methods are applied, respectively, to circuit sizing and both sizing and topology exploration, whereas in [39] particle swarm optimization is explored in both single and multi-objective approaches. A different approach is taken by Pradhan and Vemuri in [43], where the multi-objective simulated annealing (MOSA) is used.

Instead of executing circuit sizing on-the-fly, in some approaches, the non-dominated solutions are generated, prior to the design task, using the previously referred multi-objective optimization methods or variations of them for the most

relevant tradeoff and then, the suitable solution is selected from the already sized solutions [44–47].

From the study of analog circuit sizing and optimization approaches proposed by the scientific community recently, it is clear that there is not a specific trend toward a single algorithm, but many were experimented with. In the next section, the summary of the surveyed approaches is presented, and finally the objectives for this work are refined, namely the selection of the optimization kernels to be initially included in the platform.

2.2 Optimization Techniques Applied to Analog Circuit Sizing

The analog sizing tools approaches surveyed are summarized in Table 2.1. In the equation-based systems, the usage of classical optimization methods is possible; however, the accuracy of the models and the derivation of such equations strongly limit the applicability. This limitation of the equation-based systems is overcome at the expense of evaluation time by using accurate circuit simulation to evaluate the performance figures being optimized.

Using the circuit simulator, methods that take advantage of some properties of the models that cannot be used, leading, as seen, to the usage of stochastic heuristic optimization techniques. From the approaches that were surveyed, the most common stochastic algorithms were based on simulation annealing and genetic/evolutionary approaches, with some of the latest implementations considering particle swarm optimization and ant colony methods.

2.2.1 Selection of Optimization Methods

This work is in the scope of circuit sizing which considers electric simulation to evaluate the circuits' performance, as illustrated in Fig. 2.2. The generality of the approach is increased and the setup time for new circuits is decreased. However, the relation between the performance figures and the design variables becomes unknown, making the usage of classic optimization methods inappropriate, as seen in the surveyed simulation-based systems where almost all consider heuristic optimization methods.

In AIDA-C, the circuit sizing and optimization problem, which will be described in detail in Chap. 3, is modeled as a multi-objective multi-constraint optimization problem. In this context, special relevance is given to multi-objective algorithms. Historically, both SA and GA have been used intensively, in this sense, it is natural to consider at least an evolutionary and an annealing. Given the recent experiments

Table 2.1 Summary of analog IC design automation tools for sizing and optimization

Tool/author		Design plan/ optimization method	Evaluation	Time setup/ exec.
IDAC [4]	1987	Design plan plus SA post-optimization	Equations	Months/few seconds
DELIGTH.SPICE [21]	1988	Feasible directions optimization	Simulator	Moderate/18 h
OPASYN [6]	1990	Steepest descent	Equations	2 weeks/5 min
OPTIMAN [15]	1990	SA	Equations	⊘/1 min
STAIC [7]	1992	two-step optimization	Equations	Long/2 min
Maulik et al. [8, 9]	1993	B&B, and sequential quadratic program	Equations and BSIM models	6 months/1 min
FRIDGE [23]	1994	SA	Simulator	1 h/45 min
DARWIN [17]	1995	GA	Small signal, analytical expressions	⊘/⊘
ISAID [2, 3]	1995	Qualitative reasoning + post optimization	Equations and qualitative reasoning	⊘/⊘
FASY [13, 14]	1996	SA + Gradient	Simulator	⊘/6 h
ASTRX/OBLX [16]	1996	SA	AWE equations	Few days/few seconds
Koza [31]	1997	GA	Simulator	⊘/⊘
GPCAD [11]	1998	Geometric programming	Posynomial	⊘/fast
MAELSTROM [29]	1999	GA + SA	Simulator	⊘/3, 6 h
ANACONDA [30]	2000	Stochastic pattern search	Simulator	⊘/10 h
Sripramong [32]	2002	GA	Simulator	⊘/3 days
Alpaydin [27]	2003	Evolutionary strategies + SA	Fuzzy + NN trained with Simulator	⊘/45 min
Shoou-Jin [48]	2006	GA	Equations	⊘/⊘
Barros [1, 25, 26]	2006	GA	Simulator	⊘/20 min
Castro-Lopez [24]	2008	SA + Powell's method	Simulator	⊘/25 min
Santos-Tavares [28]	2008	GA	Simulator	⊘/⊘
MOJITO [42]	2009	NSGA-II	Simulator	⊘/<7 days
Pradhan [43]	2009	Multi-objective SA	Layout aware MNA models	⊘/16 min
Matsukawa [10]	2009	Convex optimization	Convex functions	⊘/⊘
Cheng [22]	2009	SA	Equations	⊘/<1 h
Hongying [33]	2010	GA with VDE	Simulator	⊘/⊘
Fakhfakh [39]	2010	Multi-objective PSO	Equations	⊘/<1 min

(continued)

Table 2.1 (continued)

Tool/author		Design plan/ optimization method	Evaluation	Time setup/ exec.
Kuo-Hsuan [12]	2011	Convex optimization	Posynomial	⦸/1 h
		Stochastic fine tuning	Simulator	
Kamisetty et al. [37]	2011	PSO		⦸/⦸
Benhala et al. [36]	2012	ACO	Equation	⦸/<1 min
Roca et al. [46]	2012	NSGA-II	Simulator	⦸/⦸
Gupta and Gosh [35]	2012	ACO	Simulator	⦸/<2 h
Kumar and Duraiswamy [38]	2012	PSO	Simulator	⦸/⦸
Genom-POF [40, 41]	2012	NSGA-II	Simulator	⦸/<1 h

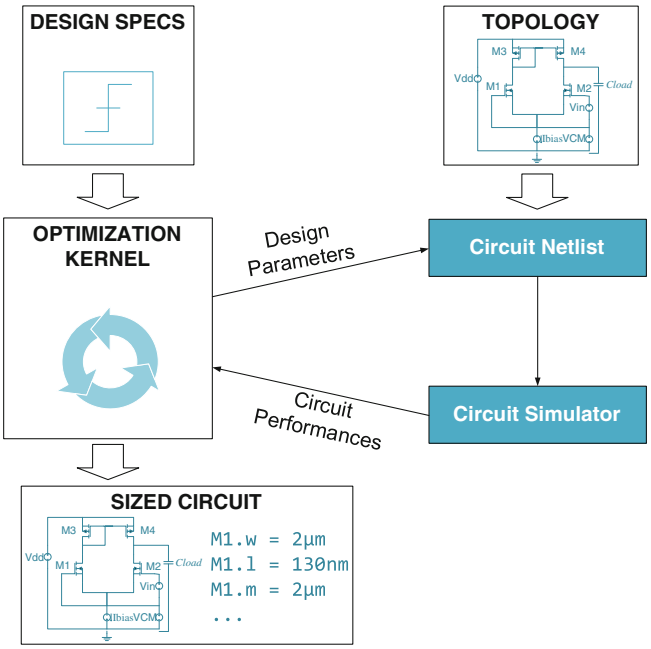

Fig. 2.2 Automatic optimization-based circuit sizing using simulator

with swarm intelligence in this domain, and to broad the scope of the implemented framework, this class of methods should also be considered.

From a brief perusal of the multi-objectives implementations, the ideas like non-sorted domination or crowding distance (further described in Chap. 3) presented in NSGA-II are reused by several other methods, as that the advantages of the inclusion of NSGAII in the framework are clear. Given the usage of SA, some sort

of multi-objective SA should also be considered. In terms of the swarm intelligence algorithms, both ACO and PSO have been applied to circuit sizing. Because MOPSO is already found in the literature and the unnatural application to real valued problems of the path finding ideas of the ACO, MOPSO will be considered.

2.3 Conclusions

In this survey, several ADA tools were presented and analyzed to better understand the advantages, and drawbacks, that can be improved in the future. It was also possible to identify that a wide range of optimization techniques are considered in this domain and new ones are always being introduced.

In this work, AIDA-CMK improves AIDA-C by adding a flexible and systematic manner to try and experiment new optimization techniques, so that further improvements to the automation of analog circuits design, namely in the circuit sizing and optimization, can be implemented more efficiently. The trends in optimization methods were also surveyed, showing a predominance of the multi-objective approaches in recent works, and the presented study was used to select a set of methods that will be considered initially to demonstrate the proposed solution.

References

1. Barros, M.F.M., Guilherme, J.M.C., Horta, N.C.G.: Analog circuits and systems optimization based on evolutionary computation techniques. Springer, Berlin (2010)
2. Makris, C.A., Toumazou, A.C.: Analog IC design automation. II. Automated circuit correction by qualitative reasoning. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **14**(2), 239–254 (1995)
3. Toumazou, C., Makris, C.A.: Analog IC design automation. I. Automated circuit generation: new concepts and methods. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **14**(2), 218–238 (1995)
4. Degrauwe, M.G.R., Nys, O., Dijkstra, E., Rijmenants, J., Bitz, S., Goffart, B.L.A.G., Vittoz, E. A., Cserveny, S., Meixenberger, C., Stappen, G.V.D., Oguey, H.J.: IDAC: an interactive design tool for analog CMOS circuits. IEEE J. Solid-State Circuits **22**(6), 1106–1116 (1987)
5. Horta, N.: Analogue and mixed-signal systems topologies exploration using symbolic methods. Analog Integr. Circ. Sig. Process **31**(2), 161–176 (2002)
6. Koh, H.Y., Sequin, C.H., Gray, P.R.: OPASYN: a compiler for CMOS operational amplifiers. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **9**(2), 113–125 (1990)
7. Harvey, J.P., Elmasry, M.I., Leung, B.: STAIC: an interactive framework for synthesizing CMOS and BiCMOS analog circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **11**(11), 1402–1417 (1992)
8. Maulik, P.C., Carley, L.R., Allstot, D.J.: Sizing of cell-level analog circuits using constrained optimization techniques. IEEE J. Solid-State Circuits **28**(3), 223–241 (1993)

9. Maulik, P.C., Carley, L.R., Rutenbar, R.A.: Integer programming based topology selection of cell-level analog circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **14**(4), 401–412 (1995)
10. Matsukawa, K., Morie, T., Tokunaga, Y., Sakiyama, S., Mitani, Y., Takayama, M., Miki, T., Matsumoto, A., Obata, K., Dosho, S.: Design methods for pipeline and delta-sigma A-to-D converters with convex optimization. In: Asia and South Pacific Design Automation Conference, 2009
11. Hershenson, M.D.M., Boyd, S.P., Lee, T.H.: GPCAD: a tool for CMOS op-amp synthesis. In: IEEE/ACM International Conference on Computer-Aided Design, San Jose, 1998
12. Kuo-Hsuan, M.: Po-Cheng, P., Hung-Ming, C.: Integrated hierarchical synthesis of analog/RF circuits with accurate performance mapping. In: International Symposium on Quality Electronic Design, Santa Clara, 2011
13. Torralba, A.J., Chavez, J., Franquelo, L.G.: Fuzzy-logic-based analog design tools. Micro, IEEE **16**(4), 60–68 (1996)
14. Torralba, A., Chavez, J., Franquelo, L.G.: FASY: a fuzzy-logic based tool for analog synthesis. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **15**(7), 705–715 (1996)
15. Gielen, G.G.E., Walscharts, H.C.C., Sansen, W.M.C.: Analog circuit design optimization based on symbolic simulation and simulated annealing. IEEE J. Solid-State Circuits **25**(3), 707–713 (1990)
16. Ochotta, E.S., Rutenbar, R.A., Carley, L.R.: Synthesis of high-performance analog circuits in ASTRX/OBLX. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **15**(3), 273–294 (1996)
17. Kruiskamp, W., Leenaerts, D.: DARWIN: CMOS opamp synthesis by means of a genetic algorithm. In: Design Automation Conference, 1995
18. Doboli, A., Dhanwada, N., Nunez-Aldana, A., Vemuri, R.: A two-layer library-based approach to synthesis of analog systems from VHDL-AMS specifications. ACM Trans. Des. Autom. Electron. Syst. **9**(2), 238–271 (2004)
19. Nagel, L.W.: SPICE2: a computer program to simulate semiconductor circuits. EECS Department, University of California, Berkeley, 1975
20. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220** (4598), 671–680 (1983)
21. Nye, W., Riley, D., Sangiovanni-Vincentelli, A., Tits, A.: DELIGHT.SPICE: an optimization-based system for the design of integrated circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **7**(4), 501–519 (1988)
22. Cheng-Wu, L., Pin-Dai, S., Ya-Ting, S., Soon-Jyh, C.: A bias-driven approach for automated design of operational amplifiers. In: International Symposium on VLSI Design, Automation and Test, Hsinchu, 2009
23. Medeiro, F., Fernandez, F., Dominguez-Castro, R., Rodriguez-Vazquez, A.: A statistical optimization-based approach for automated sizing of analog cells. In: International Conference Computer-Aided Design, 1994
24. Castro-Lopez, R., Guerra, O., Roca, E., Fernandez, F.: An integrated layout-synthesis approach for analog ICs. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **27**(7), 1179–1189 (2008)
25. Barros, M., Guilherme, J., Horta, N.: Analog circuits optimization based on evolutionary computation techniques. Integr. VLSI J. **43**(1), 136–155 (2010)
26. Barros, M., Guilherme, J., Horta, N.: GA-SVM feasibility model and optimization kernel applied to analog IC design automation. In: ACM Great Lakes symposium on VLSI, Stresa-Lago Maggiore, 2007
27. Alpaydin, G., Balkir, S., Dundar, G.: An evolutionary approach to automatic synthesis of high-performance analog integrated circuits. IEEE Trans. Evol. Comput. **7**(3), 240–252 (2003)
28. Santos-Tavares, R., Paulino, N., Higino, J., Goes, J., Oliveira, J.P.: Optimization of multi-stage amplifiers in deep-submicron CMOS using a distributed/parallel genetic algorithm. In: International Symposium on Circuits and Systems, Seattle, 2008

29. Krasnicki, M., Phelps, R., Rutenbar, R., Carley, L.: MAELSTROM: efficient simulation-based synthesis for custom analog cells. In: Design Automation Conference, New Orleans, 1999
30. Phelps, R., Krasnicki, M., Rutenbar, R., Carley, L., Hellums, J.: Anaconda: simulation-based synthesis of analog circuits via stochastic pattern search. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **19**(6), 703–717 (2000)
31. Koza, J.R., Bennett, F.I., Andre, D., Keane, M.A., Dunlap, F.: Automated synthesis of analog electrical circuits by means of genetic programming. IEEE Trans. Evol. Comput. **1**(2), 109–128 (1997)
32. Sripramong, T., Toumazou, C.: The invention of CMOS amplifiers using genetic programming and current-flow analysis. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **21**(11), 1237–1252 (2002)
33. Hongying, Y., Jingsong, H.: Evolutionary design of operational amplifier using variable-length differential evolution algorithm. In: International Conference on Computer Application and System Modeling, Taiyuan Shanxi, 2010
34. Chu, S.-C., Huang, H.-C., Roddick, J.F., Pan, J.-S.: Overview of algorithms for swarm intelligence. In: Computational Collective Intelligence. Technologies and Applications, pp. 28–41. Springer, Berlin Heidelberg (2011)
35. Gupta, H., Ghosh, B.: Analog Circuits Design Using Ant Colony Optimization. Int. J. Electron. Comput. Commun. Technol. **2**(3), 9–21 (2012)
36. Benhala, B., Ahaitouf, A., Fakhfakh, M., Mechaqrane, A.: New Adaptation of the ACO Algorithm for the Analog Circuits Design Optimization. Int. J. Comput. Sci. Issues **9**(3), 360–367 (2012)
37. Kamisetty, S., Garg, J., Tripathi, J., Mukherjee, J.: Optimization of analog RF circuit parameters using randomness in particle swarm optimization. In: World Congress on Information and Communication Technologies, 2011
38. Kumar, P.P., Duraiswamy, K.: An optimized device sizing of analog circuits using particle swarm optimization. J. Comput. Sci. **8**(6), 930–935 (2012)
39. Fakhfakh, M., Cooren, Y., Sallem, A., Loulou, M., Siarry, P.: Analog circuit design optimization through the particle swarm optimization technique. Analog Integr. Circ. Sig. Process **63**(1), 71–82 (2010)
40. Lourenço, N., Horta, N.: GENOM-POF: multi-objective evolutionary synthesis of analog ICs with Corners validation. In: Genetic and Evolutionary Computation Conference, Philadelphia, 2012
41. Lourenço, N., Martins, R., Barros, M., Horta, N.: Analog circuit design based on robust POFs using an enhanced MOEA with SVM models. In: Analog/RF and Mixed-Signal Circuit Systematic Design, pp. 149–167. Springer, Berlin (2013)
42. McConaghy, T., Palmers, P., Steyaert, M., Gielen, G.: Trustworthy genetic programming-based synthesis of analog circuit topologies using hierarchical domain-specific building blocks. IEEE Trans. Evol. Comput. **15**(4), 557–570 (2011)
43. Pradhan, A., Vemuri, R.: Efficient synthesis of a uniformly spread layout aware pareto surface for analog circuits. In: International Conference on VLSI Design, New Delhi, 2009
44. Deniz, E., Dundar, G.: Hierarchical performance estimation of analog blocks using Pareto Fronts. In: Ph.D. Research in Microelectronics and Electronics, 2010
45. Castro-Lopez, R., Roca, E., Fernandez, F.V.: Multimode Pareto fronts for design of reconfigurable analogue circuits. Electron. Lett. **45**(2), 95–96 (2009)
46. Roca, E., Velasco-Jiménez, M., Castro-López, R., Fernández, F.V.: Context-dependent transformation of Pareto-optimal performance fronts of operational amplifiers. Analog Integr. Circ. Sig. Process **73**(1), 65–76 (2012)
47. Gielen, G., McConaghy, T., Eeckelaert, T.: Performance space modeling for hierarchical synthesis of analog integrated circuits. In: Design Automation Conference, 2005
48. Shoou-Jinn, C., Hao-Sheng, H., Yan-Kuin, S.: Automated passive filter synthesis using a novel tree representation and genetic programming. IEEE Trans. Evol. Comput. **10**(1), 93–100 (2006)

Chapter 3
AIDA-CMK: AIDA-C with MOO Framework

Abstract This chapter explains the circuit optimization tool, AIDA-C, and the changes proposed in this work to enhance the tool with multiple algorithms, leading to AIDA-CMK.

Keywords Analog IC design · Automatic circuit sizing · Circuit optimization · Electronic design automation · Computer-aided design

3.1 AIDA-C Framework

AIDA-C stems from GENOM-POF [1] and is part of the AIDA [2] design automation framework, illustrated in Fig. 3.1, that implements the complete analog IC design flow from device sizing to layout generation. AIDA-C performs the analog circuit sizing and optimization part of the flow, addressing robust design requirements by considering extreme process, voltage, and temperature (PVT) corner conditions together with the use of the industrial grade circuit simulators, HSPICE® [3] and ELDO® [4], for accurate circuit performance evaluation. AIDA-C can also use AIDA-L's floorplanner to add geometrical layout measures (e.g., total area, device area, aspect ratio, etc.,) to the set of the circuit's performance figures that are considered during optimization.

Finally, the layout generator AIDA-L, previously known as LAYGEN-II [5], inputs the device sizes and floorplan template to generate the corresponding layout by placing and routing all the devices, completing the design flow. A final validation step is done using the physical verification tool CALIBRE® [6].

In AIDA-C, circuit sizing and optimization is implemented as a multi-objective multi-constraint optimization problem, originally supported by the multi-objective evolutionary algorithm NSGA-II. The multi-objective optimization problem is defined in (3.1), where, x is a vector of N optimization variables, $g_j(x)$ one of the J constraints, and $f_m(x)$ one of the M objective functions.

R. Lourenço et al., *AIDA-CMK: Multi-Algorithm Optimization Kernel Applied to Analog IC Sizing*, SpringerBriefs in Computational Intelligence,
DOI 10.1007/978-3-319-15955-3_3

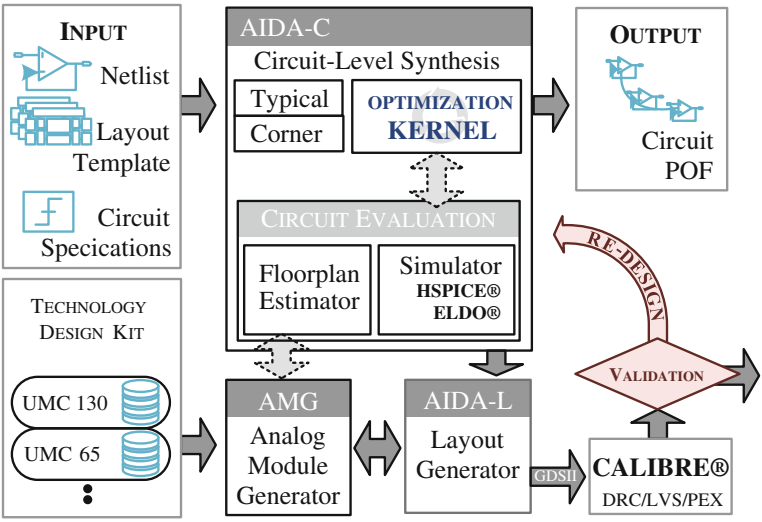

Fig. 3.1 AIDA framework

$$\begin{aligned} &\text{find } x \text{ that minimize } f_m(x) && m = 1, 2, \ldots M \\ &\text{subject to } g_j(x) \geq 0 && j = 1, 2, \ldots J \\ &x_i^{\mathrm{L}} \leq x_i \leq x_i^{\mathrm{U}} && i = 1, 2, \ldots N \end{aligned} \tag{3.1}$$

Given the multi-objective nature of the sizing method, the output is not one solution but a set of solutions all compliant with the design specifications. The optimizer's output is a set of Pareto nondominated solutions or Pareto front. Pareto dominance states that one point in the solution space, A, is not dominated by another point B, if $\exists m: f_m(A) < f_m(B)$. Figure 3.2 depicts a Pareto front, illustrating the concept, where solutions A and B are nondominated and both dominate solution point C.

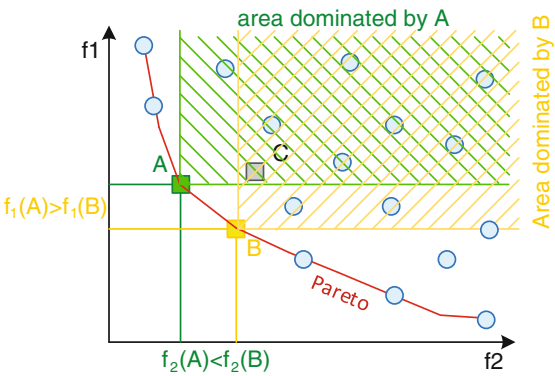

Fig. 3.2 Pareto front illustration

AIDA-C targets the sizing of the devices in analog circuits using state-of-the-art and innovative techniques. The focus of this work is to enhance the optimization kernel with an abstraction layer that permits the easy inclusion of new optimization techniques aside the NSGA-II, which was originally supported. As stated in Chap. 2, besides the definition and implementation of the framework, default implementations of MOPSO and MOSA methods are also provided and applied to circuit optimization. The new tool AIDA-CMK, after the implementation of the optimization kernel framework, is shown in Fig. 3.3.

Additionally, the algorithm implementations share a common interface with AIDA-CMK and between themselves, easing the intermingling of tentative solutions between techniques in order to, not only use the different approaches by themselves, but also ease hybridization. To explore this feature a Hybrid method that combines the previously referred optimization kernels is also implemented and explained in Sect. 3.6.

The definition in (3.1) is used to create an abstraction layer between the optimization method and the circuit being optimized, where the evaluation of the circuit performance is done using the circuit simulator. In AIDA-C geometric and performance figures that depend on the physical layout representation of the circuit are considered using AIDA-L to generate the layout and extract them. However, in the scope of this work, as the algorithms to be implemented do not depend on the tool or tools used to measure the circuit performance, layout dependent performance measures are not considered.

The next section describes how the circuit sizing problem is mapped into the multi-objective optimization problem as defined in (3.1), and the following sections will describe with more detail the optimization kernels considered in this first implementation.

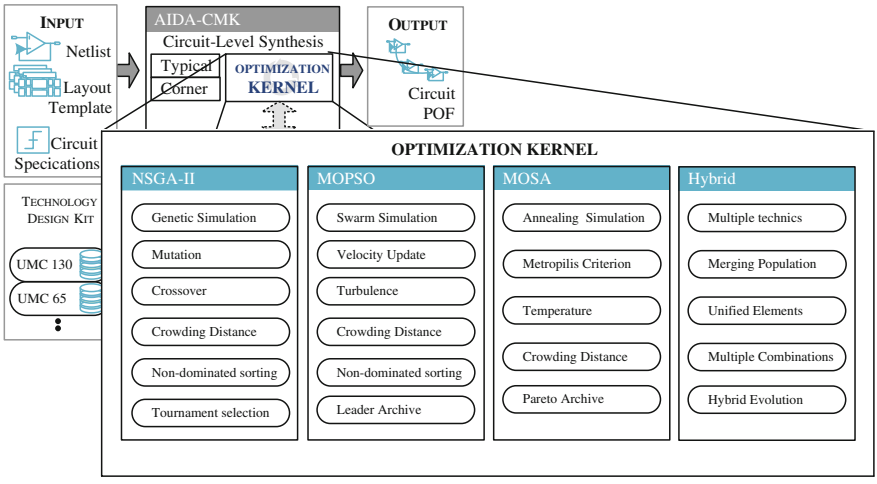

Fig. 3.3 AIDA-CMK optimization algorithms

3.2 Circuit Sizing as Multi-objective Optimization Problem

In this section, the procedure used to convert the analog IC designer inputs to the problem formulation shown in (3.1) is described. The multi-objective optimization, as defined in (3.1), to the circuit sizing problem, the inputs from the designer, which are not provided as the tuple $\{x, f, g\}$, need to be properly mapped.

The simple differential amplifier from Fig. 3.4 will be used to illustrate the procedure, where Tables 3.1, 3.2, and 3.3 show the circuit parameters, design objectives, and target specifications, respectively.

When considering only typical case simulation, the design objectives being minimized are used directly as one of the $f_m(x)$, and the ones being maximized are multiplied by -1. The design constraints are normalized and multiplied by -1, if necessary, according to (3.2), where, p_i is the measured circuit characteristic and P_i is the correspondent acceptable limit. The circuit parameters are used as ranged design variables and define the search space.

$$f_m(x) = \begin{cases} p_m & \text{when minimizing } p_m \\ -p_m & \text{when maximizing } p_m \end{cases}$$

$$g_i(x) = \begin{cases} \frac{p_i - P_i}{|P_i|} & \text{when the constraint is } p_i \geq P_i \\ p_i & \text{when the constraint is } p_i \geq P_i \text{ and } P_i = 0 \\ -p_i & \text{when the constraint is } p_i \leq P_i \text{ and } P_i = 0 \\ \frac{P_i - p_i}{|P_i|} & \text{when the constraint is } p_i \leq P_i \end{cases} \tag{3.2}$$

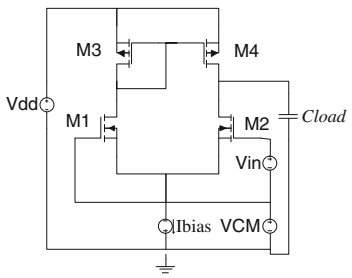

Fig. 3.4 Differential amplifier

Table 3.1 Parameter ranges for the differential amplifier

Variables	W1	W2	L1	L2	Ib
Maximum	400.0e-6	400.0e-6	15.0e-6	15.0e-6	500.0e-6
Minimum	1.0e-6	1.0e-6	0.35e-6	0.35e-6	100.0e-6

Table 3.2 Objectives for the example in differential amplifier

Performance	Target	Units	Description
A0	Maximize	dB	Gain DC
Gbw	Maximize	MHz	Unit-gain frequency

Table 3.3 Design specifications for the example in differential amplifier

Performance	Target	Units	Description
Gbw	≥35	MHz	Unit-gain frequency
Pm	65 ≤ pm ≤ 90	Degree	Phase margin
vov_m1	50 ≤ vov_m1 ≤ 200	mV	Vgs–Vt
vov_m2	50 ≤ vov_m2 ≤ 200	mV	Vgs–Vt
vov_m3	100 ≤ vov_m3 ≤ 300	mV	Vgs–Vt
vov_m4	100 ≤ vov_m4 ≤ 300	mV	Vgs–Vt
delta_m1	≥50	mV	Vds–Vdsat
delta_m2	≥50	mV	Vds–Vdsat
delta_m3	≥50	mV	Vds–Vdsat
delta_m4	≥50	mV	Vds–Vdsat

Table 3.4 $f_m(x)$ and $g_j(x)$ for the differential amplifier example

Constraints	$g_0(x) = \frac{gbw}{35 \times 10^6} - 1$	$g_1(x) = \frac{pm}{65} - 1$	$g_2(x) = 1 - \frac{pm}{90}$
	$g_3(x) = \frac{vov\_m1}{50 \times 10^{-3}} - 1$	$\cdots$	$g_{15}(x) = \frac{delta\_m4}{50 \times 10^{-3}} - 1$
Objectives	$f_0(x) = -gain\_dc$	$f_1(x) = -gbw$	–

Table 3.4 illustrates the objective and constraint functions for the differential amplifier circuit in Fig. 3.4. Modeling the problem in this manner, the circuit is evaluated only for nominal conditions; therefore it requires less simulations, i.e., is faster. Despite the output does not consider the limitations imposed by the extreme variations of process and environment parameters, it is useful to the circuit designer to perform tradeoff analysis.

A critical problem in analog IC design is Process, supply Voltage and Temperature variability (PVT). This phenomenon affects devices in different chips and also devices within the same chip, i.e., devices designed to be equal are different after production due to manufacturing mismatches and are solved by robust circuit design. In order to verify if the design is robust, i.e., the vast majority of the fabricated circuits will work according to the specifications, special techniques are employed. One such technique is PVT corner simulations. PVT Corners is a worst-case approach where the circuit is simulated over multiple combinations of extreme process parameter variations, power supply, temperature, etc., that lead to the worst-case performance. Figure 3.5 illustrates eight corner cases obtained by considering three values for power supply, operating temperature, and process parameters.

To include these effects in the optimization, the design is evaluated considering all the considered corners, i.e., for each evaluation, the circuit is simulated once for each corner case, this makes the execution slower when compared to typical, but the output circuits are ensured to be feasible in all tested corner conditions. To handle the multiple corners, the objective and constraint functions are modified using (3.3), where, C is the number of corners, and $f_m^c(x)$ and $g_j^c(x)$ are respectively the

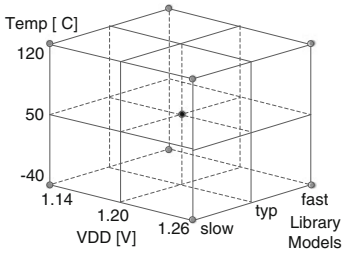

Fig. 3.5 Corner cases

objective $f_m(x)$, and the constraint $g_j(x)$, as defined for the typical case evaluated in corner case c.

$$\hat{f}_m(x) = \max_{c=1,2,\ldots,C} \left(f_m^c(x)\right)$$
$$\hat{g}_j(x) = \sum_{c=1}^{C} c_j^c(x) \text{ with } c_j^c(x) = \begin{cases} 0 & \text{if } g_j^c(x) \geq 0 \\ g_j^c(x) & \text{if } g_j^c(x) < 0 \end{cases} \tag{3.3}$$

In this worst-case approach, each objective, which is being minimized, evaluates to the maximum value obtained from the simulation of circuit in all the corner cases, and each constraint is evaluated as the sum of the normalized violation in all the corner cases where it is violated.

Hence, both nominal and worst-case optimization is mapped to the tuple (x, f, g). From this point on, the circuit optimization is viewed by the optimization methods as the tuple $\{x, f, g\}$. Therefore, given the faster execution and without the loss of generality, testing the algorithms is done considering only the nominal case. With this interface defined, the definition of a general interface based on the standard definition of the optimization problem would be simple; however, there is a secondary requirement for this interface.

In the AIDA framework, the AIDA-L's detailed routing is also an innovative optimization-based approach, where all the wires in the layout are evolved simultaneously, unlike the remaining state-of-the-art approaches. While the study of circuit layout is out of the scope of this work, the problem definition within the developed framework should be general enough to accommodate the complex representation of the genome in the detailed Router, so this must be considered when developing the proposed interface.

Another important feature is the possibility to define elements to be used as starting point, as it permits sequential execution of multiple optimization tasks. Two important uses of this feature: execute an optimization where the evaluation is done considering only nominal conditions and use the output of that optimization task as starting point of another that considers the corner cases; and/or execute an initial optimization using algorithms suited for exploration like the GAs, and then, execute an optimization using algorithms that are more efficient exploiting the local minima like SAs.

The multi-objective optimization kernels that are going to be implemented within the scope of this work, NSGA-II, MOPSO, MOSA, and Hybrid/Multi-Kernel algorithm are described in the next sections.

3.3 Nondominated Sorting Genetic Algorithm II (NSGA-II)

NSGA-II [7] kernel is an evolutionary optimization scheme. The principle of evolutionary computation is to mimic natural evolution. The genetic algorithm starts by generating an initial population of chromosomes, the initial parents. This first population must offer a wide diversity of genetic materials. The gene pool should be as large as possible so that any solution of the search space can be engendered but generally, the initial population is generated randomly. Then, the genetic algorithm evolves the solutions by applying the genetic operators and then selecting the next parents. The process is repeated until the convergence or ending criterion is reached. The algorithm is stopped when the population converges toward the optimal solution.

New solution vectors are obtained from the current population by the application of the genetic operators of mutation and crossover. Crossover uses genes from two population elements to generate the new elements, combining randomly selected sets of information from each of the parents into the children. Mutation is a random change in individual's genetic information in order to escape from local minima; the mutation operator introduces new information in the chromosome whereas the crossover selected the best pieces of the information present in the population genetic information.

Each chromosome has an associated value corresponding to the fitness of the solution it represents. The fitness should correspond to an evaluation of how good the candidate solution is. Selection compares each individual in the population by using a fitness function. The new individuals' fitness is evaluated and, then, they are ranked together with the parents. The fittest individuals are selected as the new parents, and the less fit discarded.

In the particular case of the NSGA-II, the algorithm pseudocode is shown in Algorithm 3.1. NSGA-II uses Pareto dominance concepts to sort the multi-objective solutions.

Algorithm 3.1 NSGA II for population size P and number of generations G

```
1   parents = P new random solutions
2   generation = 0
3   while generation < G
4     offspring = apply-operators(parents) //Mutation and Crossover
5     evaluate(offspring)
6     non-dominated-sorting(parents + offspring)
7     parents = crowding-distance-selection(parents + offspring)
8     generation++
9   return parents
```

Pareto dominance is implemented by means of ranking solutions using nondominated sorting, Algorithm 3.2, and crowding distance criterion, Algorithm 3.3, as a tie breaker for solutions with the same rank. The rank process is iterative making rank-1 elements the ones that are not dominated by any other, these elements are removed from the pool of elements being sorted and the process is repeated for the next rank, until there are no more elements to be sorted, Algorithm 3.2 illustrates this procedure. Elements with lower rank dominate the ones with higher rank.

Algorithm 3.2 Non-dominated sorting procedure for a P of size N

```
1   set F1=Ø and rank=1
2   for j=1,…,N do:
3     Frank = Frank+{P[j]}
      for k=1,…,N do:
4       if(k == j) continue
5       if P[k] dominates P[j], then remove P[j] from Frank break loop in k
6   remove Frank from P and update N = N – size(Frank)
7   if P is not empty, then set rank = rank+1 and go to 2
```

The cubic approach to compute the rank of the elements in the population was introduced to clearly describe the nondominated sorting, however, in practice, the fast nondominated sorting algorithm described in [7] is used. It first computes the dominance between all solutions, storing the set of elements that are dominated (*Sd*) and the number of elements that dominate (*d*) for each solution. Using this information, all the solutions that are not dominated (have zero elements dominating it, i.e., $d = 0$) are set to the rank 1 and removed from the elements pool decrementing *d* for all the solutions in their *Sd*s. The process is repeated for rank 2, and successively until the elements pool is empty, leading to a quadratic algorithm.

To solve ties between elements of the same rank, the crowding distance criterion is used. The crowding distance is an estimate of the density of elements. Each element with the same rank is assigned a value that relates to the distance to the closer elements. Figure 3.6 illustrates the four ranking fronts and the crowding distance of the solution B in a problem with two objectives. The crowding distance of the elements in a front is computed by iterating in the *M* objective functions, sorting the elements using each objective and for each element accumulating the

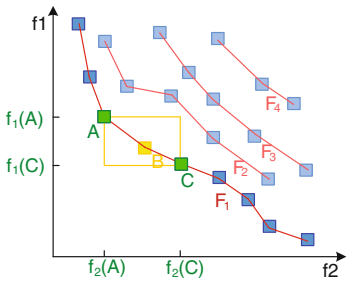

Fig. 3.6 Fronts for multiple ranks, and crowding distance for solution B

normalized value of the distance between the elements before and after in the ordered set. The boundary elements (element with smaller and higher value of each objective) are assigned with infinite value of crowding distance. The pseudocode of crowding distance computation is shown in Algorithm 3.3.

Algorithm 3.3 Crowding distance computation of front F_T of size N in a problem with M objectives

```
1   for i=1,…,N do: F_T[i].cdist=0
2   for m=1,…,M do:
3     S = F_T sorted using the value objective function f_m
4     S[1].cdist = S[N].cdist = ∞
5     for j=2,…,N-1 do:
6       S[1].cdist+=(S[i+1].f_m - S[i-1].f_m)/( f_m^max - f_m^min)
```

3.4 Multi-objective Simulated Annealing (MOSA)

The MOSA is an adaptation of the single objective simulated annealing [8] to the multi-objective case. Like evolutionary algorithms, SA is inspired by a natural phenomenon. As the algorithm name states, it simulates the cooling and annealing of liquid material. When a liquid material cools and anneals quickly, the material will solidify into a suboptimal configuration. However, if the liquid material cools slowly, the crystals within the material will solidify optimally into a state of minimum energy, i.e., ground state.

The algorithm steps are outlined in Algorithm 3.4. Despite the differences of concept, when comparing SA with stochastic hill climber, one cannot ignore the similarities. Nevertheless, in the SA the cooling schedule introduces the ability to explore broader solutions in the beginning, when the temperature is higher and performs almost like hill climber for very low temperatures. This makes cooling schedule extremely important to the performance of the algorithm.

In theory, an infinitesimal decrease of T over time leads to the global optimum solution. From a practical point of view, there are some important aspects to

Algorithm 3.4 Single Objective Simulated Annealing

```
1   T = Tmax
2   u = Random Solution
3   fu = evaluate(u)
4   while T < Tmin do:
5     v = neighbor(u)
6     fv = evaluate(v)
7     if fv < fu
8       u = v
9     else if rand(0,1) < exp((fu – fv)/(fu.T))
10      u = v
11    T = update(T)
12  return u
```

consider when devising the cooling schedule. If the initial temperature is to low or temperature drops too fast premature convergence occurs, if the initial temperature is too high or temperature drops too slowly finding the solution will take longer. This is the main tradeoff when designing the cooling schedule. One common scheduling procedure is the exponential cooling shown in (3.4).

$$T_{K+1} = \frac{T_1{}^K}{T_0} T_K \tag{3.4}$$

Because the nature of the simulated annealing is to explore the neighborhood of single tentative solutions, the adaptation to multi-objective requires changes in the structure of the search. The straightforward approach is to create a weighted combination of the objectives and find the set of Pareto optimal solutions with multiple runs of the SA with different weights; but finding the correct weights is complex. Another such adaptation can be found in [9], where the adaptation to the multi-objective case is done using an archive that stores the best solutions and the acceptance of a new solution is based on the dominance with respect to the archive not just the current solution. However, by exploring the neighborhood of just one solution at a time, the diversity of the solutions found suffers.

The implemented MOSA is shown in Algorithm 3.5, where instead of using $T_{\min}$ to control the annealed schedule, a maximum number of iterations to control the termination is used.

Algorithm 3.5 Multi-Objective Simulated Annealing

```
1   T = Tmax, iteration = 0
2   archive = P Random Solution
3   evaluate(archive)
4   pareto = non-dominated-solutions(archive)
5   while iteration < N do:
6     neighbors = neighbor-foreach-element(archive)
7     evaluate(archive)
8     foreach element,neighbor u,v in archive,neighbors
9       if v dominates u then v replace u in archive
10      else if u not-dominates v then keep u, add v to archive
11      else if rand(0,1) < exp(|f(u) – f(v)|/(|fu|.T)) then v replace u in archive
12    pareto = non-dominated-solutions(pareto + archive)
13    crowding-distance-trim(pareto)
14    crowding-distance-trim(archive)
15    T = update(T), iteration++
16  return pareto
```

The MOSA implementation follows an archive-based multi-objective simulated annealing technique, but exploring the neighborhood of the entire archive at each iteration, instead of only a single point. In this way, the diversity of the solutions explored is increased; another practical advantage is that the simulation of multiple circuits in a batch is much more efficient than simulation of a circuit at a time. The

algorithm starts with P elements in the archive instead of only one, to increase the potential for more annealing branches. The acceptance is still made by metropolis criterion and the archive is trimmed by nondominated sorting and crowding distance when its size exceeds the maximum elements permitted. A Pareto set with all the nondominated solutions found is also kept, but since the new elements are neighbors of the original solution, the Pareto archive grows very fast in solutions that are very close to each other; so a crowding distance criterion is also used to trim the Pareto set.

3.5 Multi-objective Particle Swarm Optimization (MOPSO)

Particle Swarm Optimization algorithms (PSOs) introduced by Kennedy and Eberhart in 1995 [10] are partly inspired by the behavior of large animal swarms such as schooling fish, flocking birds, or honey bees. PSO associates each particle as a candidate solution and lets them explore the search space. This technique is focused on the collective behavior of a distributed population of simple agents that interact locally with each other.

Each particle is associated with a stochastic velocity vector which indicates where the particle is moving to. The next move of each particle at a given time, illustrated in Fig. 3.7, is a stochastic combination of the velocity in the previous time instant, the direction toward the best position ever occupied by the particle, and the direction toward the best swarm positions.

In the standard model, each particle i is associated with a position (x_i) in the search space, a velocity (v_i), the position (p_i), and fitness of the best point encountered by the particle, and the rank (g) to the best particle in the swarm. The interaction between these variables is governed by the rules in (3.5), where, x is the constriction coefficient, w is the inertia weight, and p_g is the position of the best particle in the swarm. The vectors φ_1 and φ_2 are randomly generated for each particle with entries uniformly distributed between 0 and $\varphi_{1\max}$ or $\varphi_{2\max}$, respectively.

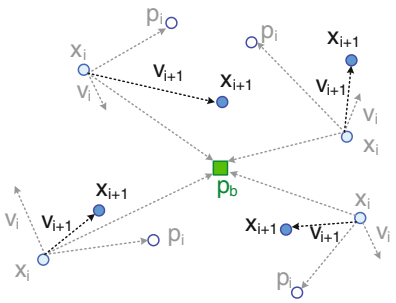

Fig. 3.7 Particle update in PSO

$$\begin{cases} \vec{x}_{i+1} = \vec{x}_i + \vec{v}_{i+1} \\ \vec{v}_{i+1} = x(w\vec{v}_i + \vec{\varphi}_{i1}(p_i - \vec{x}_i) + \vec{\varphi}_{i2}(p_g - \vec{x}_i)) \\ \vec{\varphi}_{i1} = N(0, \vec{\varphi}_{1\,\max}) \\ \vec{\varphi}_{i2} = N(0, \vec{\varphi}_{2\,\max}) \end{cases} \tag{3.5}$$

The manipulation of some of these parameters develops other variations of the standard algorithm. Controlling the velocity and the direction to the particle's best position allows the implementation of schemas for exploitation and exploration of the search space.

The implemented MOPSO follows the implementation described by [11], using external archive, turbulence, and a fully connected topology and density estimator (crowding distance). In the single objective one of the critical factors in the implementation of a PSO is the selection of the leader, in the multi-objective case the issue persists, even worsening, as there are many options to select the leader. The method selected was to randomly select a solution from the Pareto to increase the pressures for improvement, other possibilities, such as selecting a random solution from the non-dominated set of particles using crowding distance tournament between two solutions to select the leader. The pseudocode for the MOPSO is shown in Algorithm 3.6.

Algorithm 3.6 Multi-Objective Particle Swarm Optimization

```
1   step = 0
2   particles = P new random solutions
3   evaluate(particles)
4   pareto = non-dominated-solutions(particles)
5   while step < N do:
6     foreach particle p in particles
7       l = select-leader(pareto)
8       update-particle(p, l)
9       apply-turbolence(p)
10    evaluate(particles)
11    pareto = non-dominated-solutions(pareto + particles)
12    crowding-distance-trim(pareto)
13    step++
14  return pareto
```

3.6 Multi-kernel Algorithm

With the algorithms previously described developed in the platform, new optimization methods that combine their techniques can be explored. By combining and reusing the diverse strategies it is reasonable to assume that is possible to take advantage of their diverse strong points. By a careful implementation of the support framework, these algorithms can be experimented easily.

One possible combination is to use multiple algorithms in parallel, as shown in Algorithm 3.7, sharing the elements to solve the problem more efficiently. The parallel combination uses a pool of elements that is divided between each kernel and evolved using a different approach. Each time is deemed to rearrange the elements between the kernels, if *is-merge-step(step)* is true, the pool of elements is redistributed among the kernels.

The major decisions in this method are when to redistribute the elements and how the redistribution is done. A simple method to select when to rearrange the elements is to rearrange the samples at uniform periods of time, i.e., at each fixed number of steps. This was the method implemented, but more complex methods can be devised and are easy to add to the platform.

Algorithm 3.7 Parallel Multi-Kernel

```
1       step = 0
2       archive = P new random solutions
3       evaluate(archive)
4       pareto = non-dominated-solutions(archive)
5       while step < N do:
        // merge elements
9         if is-merge-step(step)
11          foreach kernel k in kernels[]
              redistribute-elements(k, archive)
        // execute optimization step
6         foreach kernel k in kernels[]
7           execute-kernel-step(k)
          archive = collect-elements(kernels[])
15        pareto = non-dominated-solutions(pareto + archive)
16        crowding-distance-trim(pareto)
18        step++
19      return pareto
```

Regarding the redistribution itself, the three methods illustrated in Fig. 3.8 were considered. In Fig. 3.8a is presented a simple version that shuffles and reassigns the elements to the different kernels taking advantage of the different exploration techniques. In Fig. 3.8b a more greedy approach is used, following the same principle of using different methods to explore the same space but selects only the best individuals using rank and crowding distance, and setting the same individuals to all the kernels. Another approach is shown in Fig. 3.8c, where the elements are sorted using some criteria and split into blocks allowing the algorithms to explore different regions of the search space.

A different method to combine the kernels is sequentially, where the results from one kernel are passed as input to the next, as shown in Algorithm 3.8. With this combination, it is possible to optimize a problem; for example, using 400 cycles of NSG-AII and then 100 cycles of MOSA to fine tune the solutions. Both multi-kernel approaches use an external Pareto set that is used to store the best solutions

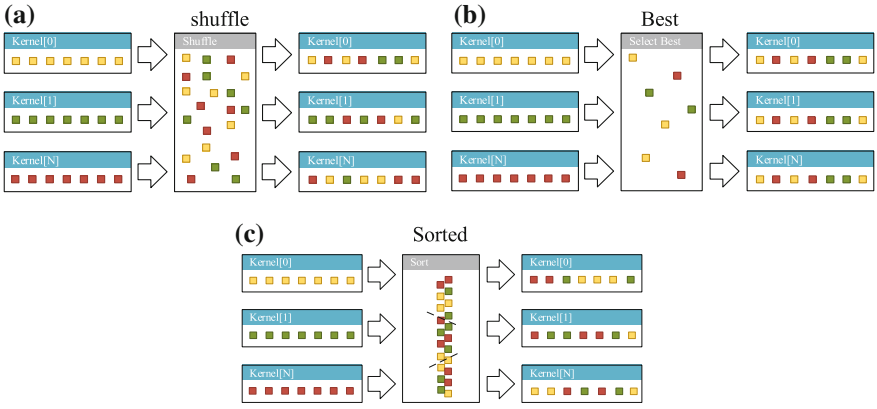

Fig. 3.8 Different methods to redistribute elements

attained during the several iterations, even if they are not the elements being considered currently in the kernels.

By combining these kernels, the framework flexibility is further enhanced and there is the possibility to find combinations of the different methods that may be better than the individual kernels. The objective in this work is to provide the infrastructure for such a study and not to conduct that study itself. Nevertheless, some of these approaches were experimented to show that the tool can now take advantage of such advanced combination of methods.

Algorithm 3.8 Sequential Multi Kernel

```
1    step = 0
2    archive = P new random solutions
3    evaluate(archive)
4    pareto = non-dominated-solutions(archive)
     k = 0
5    while step < N do:
       execute-kernel-step(kernels[k])
     // change elements
9      if is-switch-step(step)
         copy-elements(kernels[k], kernels[(k + 1)%K])
           k= (k + 1)%K]
       archive = collect-elements(kernels[k])
15     pareto = non-dominated-solutions(pareto + archive)
16     crowding-distance-trim(pareto)
18     step++
19   return pareto
```

3.7 Conclusions

In this chapter, AIDA frameworks is introduced and AIDA-C, the preceding optimization-based circuit sizing tool, is described. The architecture for the proposed tool AIDA-CMK is defined, taking into consideration how the circuit sizing is handled and the optimization problem even when considering the extra constraints introduced by PTV corners.

The multi-objective optimization kernels, NSGA-II, MOPSO, and MOSA and the two multiple kernel hybridizations implemented in AIDA-CMK are described here. Asides the infrastructure of the optimization kernels framework, which will be described in Chap. 4, AIDA-CMK supporting these algorithms are the new contributions to the AIDA framework.

References

1. Lourenço, N., Horta, N.: GENOM-POF: multi-objective evolutionary synthesis of analog ICs with corners validation. In: Genetic and Evolutionary Computation Conference, Philadelphia, 2012
2. Martins, R., Lourenço, N., Rodrigues, S., Guilherme, J., Horta, N.: AIDA: automated analog IC design flow from circuit level to layout. In: International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design, Seville, 2012
3. Synopsis Inc.: Synopsys HSPICE—accurate circuit simulation. http://www.synopsys.com (2014). Accessed 2014
4. Graphics, M.: Eldo classic—foundry certified SPICE accurate circuit simulation. http://www.mentor.com/products/ic_nanometer_design/analog-mixed-signal-verification/eldo/ (2014). Accessed 2014
5. Martins, R., Lourenço, N., Horta, N.: LAYGEN II: Analog ICs Layout Generator. Springer, Berlin (2013)
6. Graphics, M.: IC verification and signoff using calibre. http://www.mentor.com/products/ic_nanometer_design/verification-signoff/ (2014). Accessed 2014
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. Evol. Comput. IEEE Trans. **6**(2), 182–197 (2002)
8. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220** (4598), 671–680 (1983)
9. Bandyopadhyay, S., Saha, S.: Some single- and multiobjective optimization techniques. In: Unsupervised Classification, pp. 17–58. Springer, Berlin (2013)
10. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: International Conference on Neural Networks, 1995
11. Reyes-Sierra, M., Coello, C.: Multi-objective particle swarm optimizers: a survey of the state-of-the-art. Int. J. Comput. Intell. Res. **2**(3), 287–308 (2006)

Chapter 4
Multi-objective Framework Implementation

Abstract In this chapter, the details about the framework structure are presented, showing the application layers and their implementation. The structure of classes is described in detail showing their relations and the flexibility of the proposed framework.

Keywords Analog IC design · Automatic circuit sizing · Circuit optimization · Electronic design automation · Computer-aided design

4.1 Framework Structure and Design Implementation

The approach to the optimization kernel of AIDA-CMK is structured in a multilayer format considering three different layers, as illustrated in Fig. 4.1. The first layer implements the problem evaluation; the second layer implements the problem abstraction; and finally, the third layer, the optimization cycle.

In this design, the adaptability is reinforced by making the entire optimization kernel a self-contained module with well-defined interface to the exterior. One of the targets of the implementation and architecture was the improved capability of extension and maintenance of the framework code. In order to achieve those objectives, some design patterns were used such as layer, repository, observer, and model view controller.

The interface implementation between optimizer and problem follows a repository design using the optimization element as data representation common to all algorithms. This design has the advantage that every element can be used by any algorithm, but puts some effort to maintain the data representation in order to be usable by all algorithms. Also allows the optimizers to be used in another types of optimization.

The framework is centered around the abstraction layer that implements the general multi-objective problem. This interface can be instantiated as a circuit problem or as a mathematical problem, and each of those problems are evaluated

R. Lourenço et al., *AIDA-CMK: Multi-Algorithm Optimization Kernel Applied to Analog IC Sizing*, SpringerBriefs in Computational Intelligence,
DOI 10.1007/978-3-319-15955-3_4

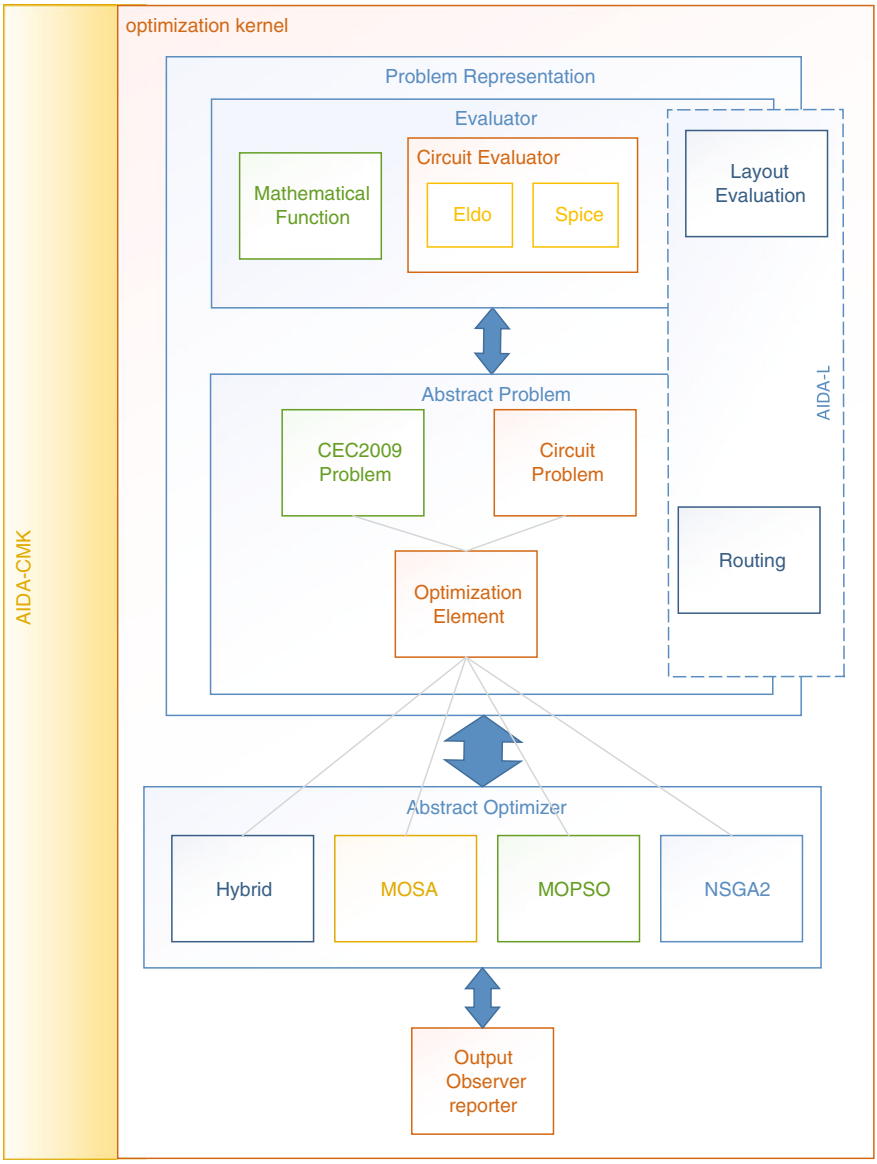

Fig. 4.1 Framework structure

using their own evaluation method, the circuit simulator or a java function, respectively. With this abstraction it is possible to test the framework with known mathematical problems in a fast and simple manner. Having the multi-objective problem in one layer and the evaluation function in another allows the problem modulation to be separated from evaluation method, making possible to switch

easily between circuit simulators or other circuit evaluation methods, while sharing the operators when the solution representations are equivalent.

By placing the responsibility of the operators' implementation in the interface between the problem and the optimizer layers, all kernels may use the same elements. This also turns the framework more versatile to new algorithm implementations by centering all the operators in the same interface. This abstraction takes advantage of optimization element generalization of the various algorithms operators, allowing the elements to be used by all algorithms and the optimizer abstraction layer makes it simple for AIDA-CMK to switch between algorithms, since all optimizers are handled equally by the optimization kernel.

Other major advantage of this design is the simplicity to implement other algorithms and strategies. The implementation of a new algorithm is done in two steps: first the enhancement of the optimization element with the operators of the new algorithm, and then the creation of another implementation of the abstract optimizer. By completing these two steps, the new kernel is ready to be applied to circuit optimization and also usable by the hybrid kernel.

The framework can also handle single objective algorithms using a vector of weights to convert a multi-objective problem in a single objective problem. However, this implementation was not explored in the circuit design problems since the circuit is converted only to a multi-objective problem. Also another implementation of the optimization elements is present in the AIDA framework, this implementation is used in the AIDA-L module of the framework to make the interface with the abstract optimizer and the routing problem. The routing element implements only the NSGA operators, so it can only be used by that kernel. However, in this design, implementing the other algorithm operators would make other kernels available to AIDA-L in the optimization of the routing.

The framework also includes a reporting module to easy visualize the algorithm outputs (POF and some statistics) and the optimization elements implements a XML handler for easy write and read the population to and from an XML format file. The output of the optimization kernel is constructed based on the observer pattern in order to create all the statistics of the optimization, passing them to AIDA application using a model view controller design.

To implement the design layers indicated above, the framework is supported by the following key abstract classes and interfaces: IEvaluator, IOptimizationSubject, and AbstractOptimizer, which are described in the following sections.

4.2 Abstract Optimization Kernel

The AbstractOptimizer class supply to the optimizer kernel, illustrated in Fig. 4.2, functions to manipulate a pool of elements, reporting functionalities, and multi-threading capabilities.

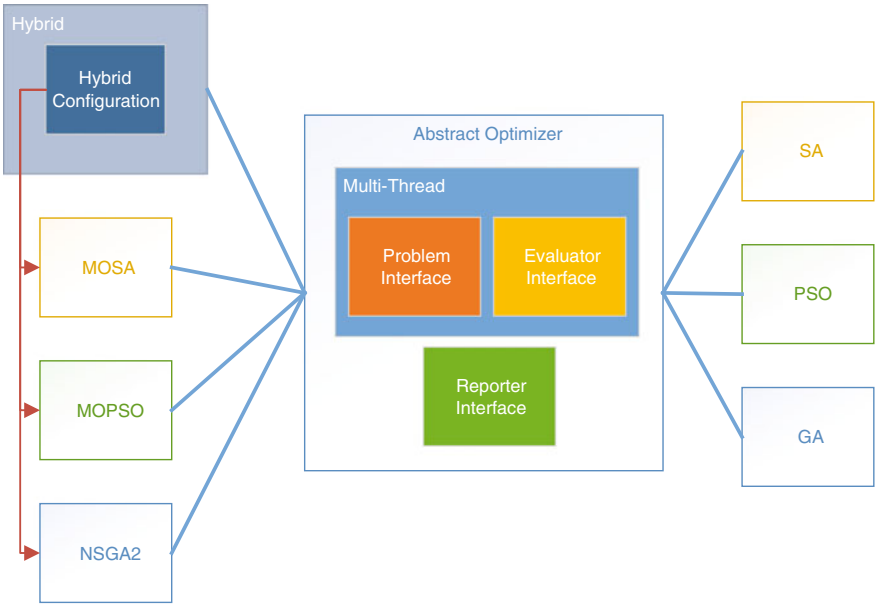

Fig. 4.2 Abstract optimizer kernel structure

The AbstractOptimizer implements the generic main loop for all the optimization methods, which is outlined in Algorithm 4.1, leaving the specific implementation to be defined in the overwritten functions.

Algorithm 4.1 NSGA II for population size P and number of generations G

```
1    initializePopulation ()
2    evaluateInitialPool()
3    updateAfterEval()
4    selection()
5    repeat {
6       applyOperators()
7       evaluatePopulation()
8       updateAfterEval()
9       selection()
10   } while (not stopCondition())
```

The AbstractOptimizer class is also responsible to maintain the output information and update the algorithm statistics such as number of evaluations and number of elements on the Pareto front. This information is stored in order to make a graphical representation of the Pareto front and its evolution interactively during the simulation that is plotted using the linux tool gnuplot.

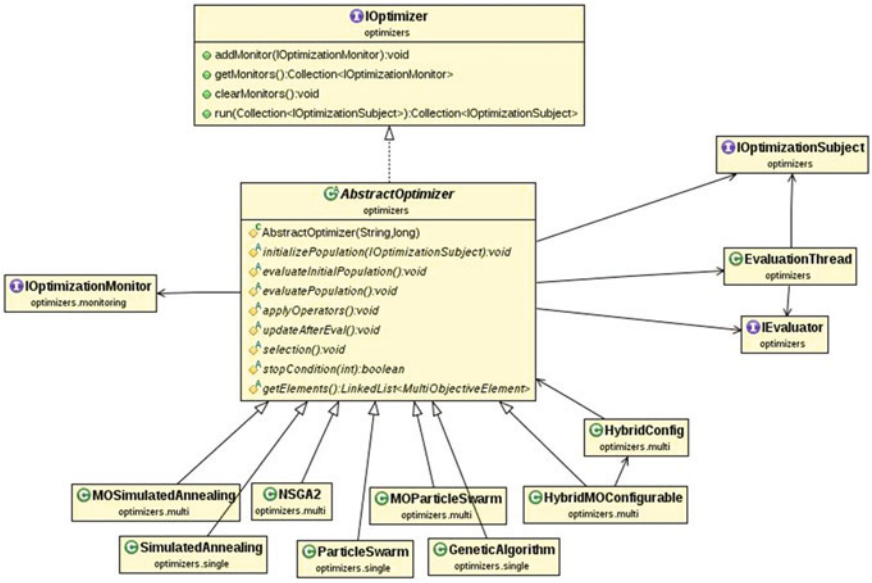

Fig. 4.3 Abstract optimizer UML diagram

The UML diagram illustrated in Fig. 4.3 describes in summary the classes and interfaces used to implement the abstract kernel. Identifying the interface, used by AIDA application, IOptimizer and the three interfaces used by the kernel to communicate with the different components, namely, IOptimizationProblem, IEvaluator, and IOptimizationMonitor. These three interfaces assure the integrity of the communication between each layer. The abstract methods of the AbstractOptimizer class and all the implemented algorithms are also present as an extension of the class.

With all the output management and multi-threading capabilities inside the AbstractOptimizer, the algorithm implementations need only to deal with specific details of the algorithm, easily allowing new developments for the extension of the existing framework. The optimization elements' capability to be evaluated based on all the objectives or a combination of them, uses a weighted vector to archive that, making it possible to implement multi-objective and single objective algorithms. In this work, the focus was on the multi-objective algorithms given the existent mapping of the circuit problems formulated as multi-objective in AIDA-CMK.

The architecture of the abstract optimizer was made to take advantage of a pool of elements. That pool can be used at any iteration by any algorithm technique, making it possible for a multi-kernel algorithm to manage the evolution of the simulation. The optimizer can evolve the elements using multiple strategies and paradigms. To do that, a list of optimization kernels must be defined in a HybridConfig settings as well as the merge strategy to be used in the algorithm. The HybridConfig also indicates the way the hybrid kernel should use the kernels, sequential or parallel, and the selection method to be used in the parallel execution when the pool is merged and divided.

4.3 Problem Representation

The problem representation, illustrated in Fig. 4.4, is another main part of the optimization kernel.

Abstracting the DoubleValueProblem, which implements the IOptimization-Subject interface, delegates to each problem implementation the responsibility of implementing or communicating the evaluation function. So, it is transparent to the Optimization Kernel of AIDA-CMK the optimization of mathematical problems or circuit problems.

This implementation is also responsible to implement the algorithms operators. DoubleValueInputProblem class is extended by the CircuitOptimizationProblem and also the mathematical problems. By doing this abstraction is also possible the use of AIDA-CMK optimization kernel in other modules of AIDA Framework namely AIDA-L, as previously referred, which also implements the IOptimizationSubject.

The UML presented in Fig. 4.5 shows the class structure of the problem, focusing the importance of the IOptimizationSubject as interface between the optimizer and the problem implementation, making possible to use the same optimizer both in circuit sizing and routing. The UML also shows the abstract DoubleValueInput-Problem creates an abstraction layer and a common base for the mathematical and

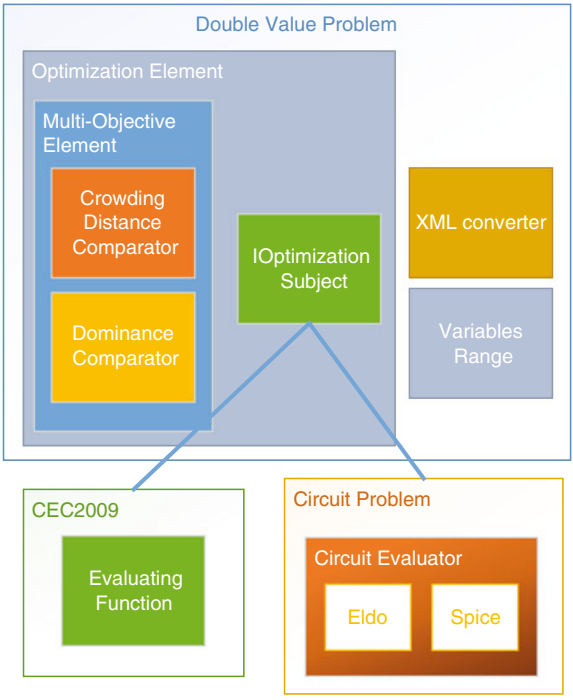

Fig. 4.4 Problem representation block diagram

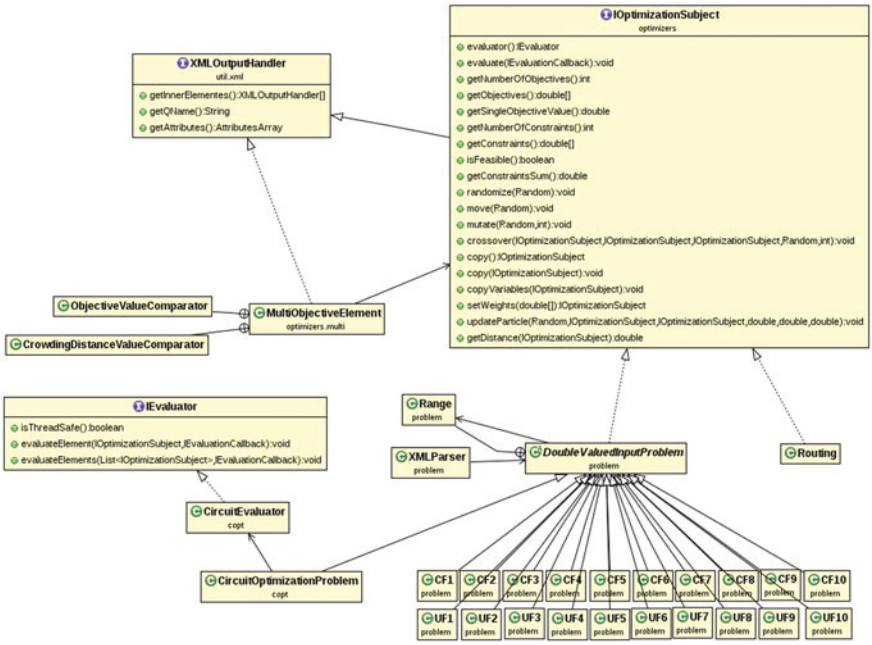

Fig. 4.5 Problem representation UML diagram

circuit problems. Another important interface and implementation presented are the IEvaluator and CircuitEvaluator. These classes are responsible for the implementation of the connection between the optimizer and the circuit simulator. As referred above, the XML interface is also presented. This interface ensures the coherent handling of all the data present both in the optimizationElement and MultiObjectiveElement.

The optimization elements interface is defined by the IOptimizationSubject. This interface is implemented by DoubleValuedInputProblem and Routing. The benchmark problems and circuit problems then extended the DoubleValuedInputProblem. Because the multi-objective algorithms need more information on each element, the MultiObjectiveElement was created to accommodate and manipulate the needed information such as crowding distance and dominance. The MultiObjectiveElement has an IOptimizationSubject element inside to represent the problem to be optimized either as a mathematical or circuit problem.

The CircuitOptimizationProblem is responsible to map the circuit as a multi-objective problem and the CircuitEvaluator to implement the interface to the circuit simulator. This level of abstraction makes the framework very versatile because it disassociates the implementation of the evaluation function from the algorithm that uses that function, making easy to change the method used to evaluate the circuit by changing the circuit simulator.

Another important feature of the problem representation is the ability to be converted from and to XML, making the elements capable of being saved in a persistent form on the hard disk during the simulation, and also, recover the simulation from a saved checkpoint. This is important in the circuit optimization because the simulation is a time consuming task and failures can occur, so a recuperation mechanism is an important feature to have.

4.4 Output and Reporting

The framework also has a reporting and graphical module. This module is structured as illustrated in Fig. 4.6 and takes advantage of the observer pattern to store and print the simulation data in a graphical form, making it possible to observe the evolution of the simulation using GnuPlot. The output module can also print boxplot graphics given a group of simulations.

As this framework is part of AIDA, the optimization must have interface with AIDA frontend, as illustrated in Fig. 4.8. This output reporting module is very important to maintain the model view controller pattern used by AIDA. The monitor is instantiated by two components as illustrated in Fig. 4.7, the ResultsPanel and MultiObjectiveReporter, both used in AIDA application to show the simulation results (A) and graphics (C), respectively illustrated in Fig. 4.8.

Additionally, several controllers where implemented to configure the kernels. Those controllers are used in the dialog that set the algorithm parameters, as shown in Fig. 4.8(B).

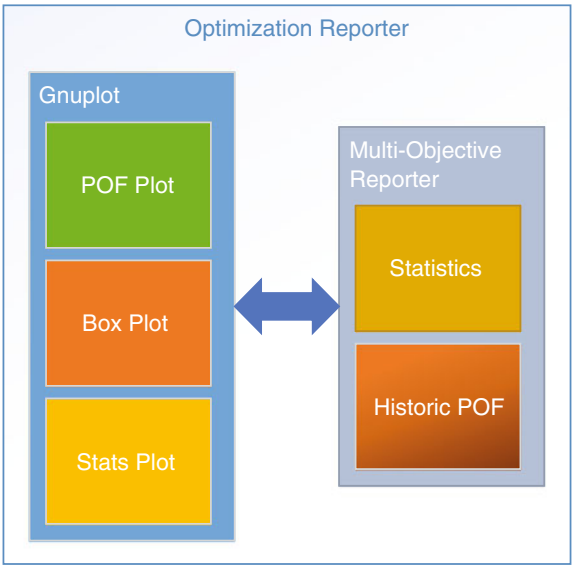

Fig. 4.6 Output and reporting block diagram

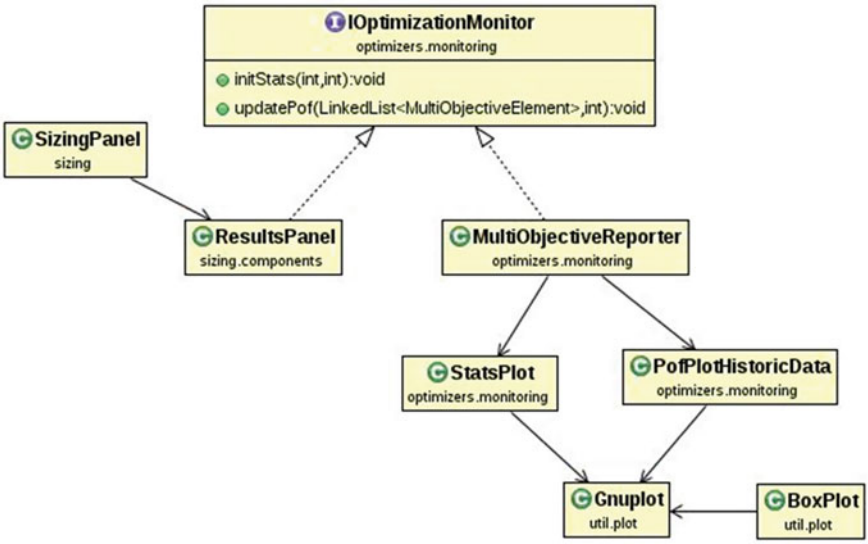

Fig. 4.7 Output and reporting UML

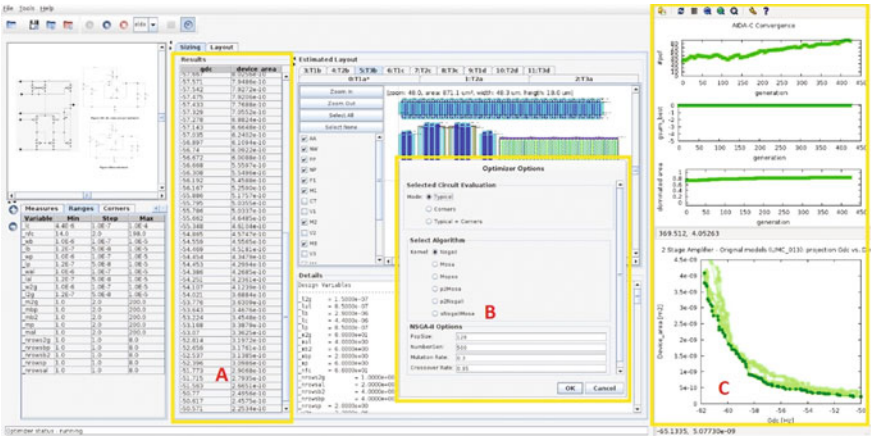

Fig. 4.8 AIDA frontend screenshot

4.5 Conclusions

In this chapter, the framework design implementation is described. The design and class structure of the extensibility capabilities of the framework are showed, indicating the advantages of each design pattern chosen to implement the framework.

Chapter 5
Kernel Validation Using CEC2009 Benchmarks

Abstract In this chapter the implemented optimization kernels are applied to mathematical problems to evaluate their performance and empirically tune the algorithms' parameters with known functions. In addition, some examples of hybrid combinations are presented. These are not exhaustive tests, and should be considered a starting point for the future research on the combinations of optimization strategies applied to analog circuit.

Keywords Optimization · Combination of optimization techniques

5.1 Problem Definition

To test and empirically tune the algorithms with known functions, some of the CEC2009 competition [1] problems were considered. In the scope of this work and because the circuit is mapped as a constrained multi-objective problem usually with two objectives, the two objective constrained problems, CF1–CF7, were selected, and the corresponding Pareto fronts are illustrated in Fig. 5.1.

5.2 Evaluation of the Single-Kernel Methods

To verify the behavior of the implemented algorithms and tune the algorithm parameters, several executions were conducted for the problems defined previously. For the initial executions the number of evaluations was selected to be around 300,000 (as in the CEC2009 competition described in [1]) and n, the number of variables, is set to 4 for all the problems.

R. Lourenço et al., *AIDA-CMK: Multi-Algorithm Optimization Kernel Applied to Analog IC Sizing*, SpringerBriefs in Computational Intelligence,
DOI 10.1007/978-3-319-15955-3_5

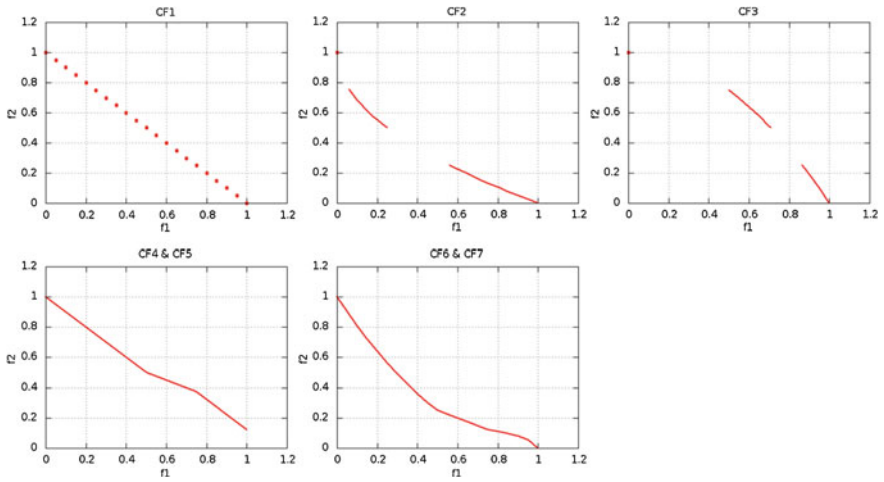

Fig. 5.1 CEC2009 Optimal Pareto fronts: CF1–CF7

The parameters of the Single Kernel Methods were set to values that are common in the literature and an empirical tuning was done. A more extensive study of the parameters' influence by conducting systematic sweeps or parameter combination sampling to fine-tune the parameters could have been done, but the fine-tuning of specific algorithms is not the objective of this work.

Using these parameters several simulations were executed for the CEC2009 CF1–CF7 problems with 4 variables. The number of evaluations considered was 294,400, leading to a population size of 128 and 2,300 generation in the NSGAII, equivalently a swarm size of 128 and 2,300 steps for the MOPSO and archive size of 128 and 2,300 iterations for the MOSA. The attained results are illustrated in Fig. 5.2. From the analysis of the plots it is clear that all the algorithms' performance in these problems is similar.

In order to do the tests in conditions closer to that of the circuit problems, normally defined with 20–30 variables, the same problems were (re)defined with the number of variables set to 30 and (re)simulated in the same conditions.

The result of these simulations is shown in Fig. 5.3. The analysis of the obtained fronts shows a notorious degradation of the MOPSO performance, when dealing with problems of larger dimension. The degradation in CF1 and CF2 is not that large, but is notorious for the other problems (CF3–CF7).

Regarding both MOSA and NSGAII the performance looks similar in Fig. 5.3. However, if the MOSPO is removed, a closer look shows that the implemented MOSA greatly outperforms the NSGA-II in CF3–CF7, as illustrated in Fig. 5.4.

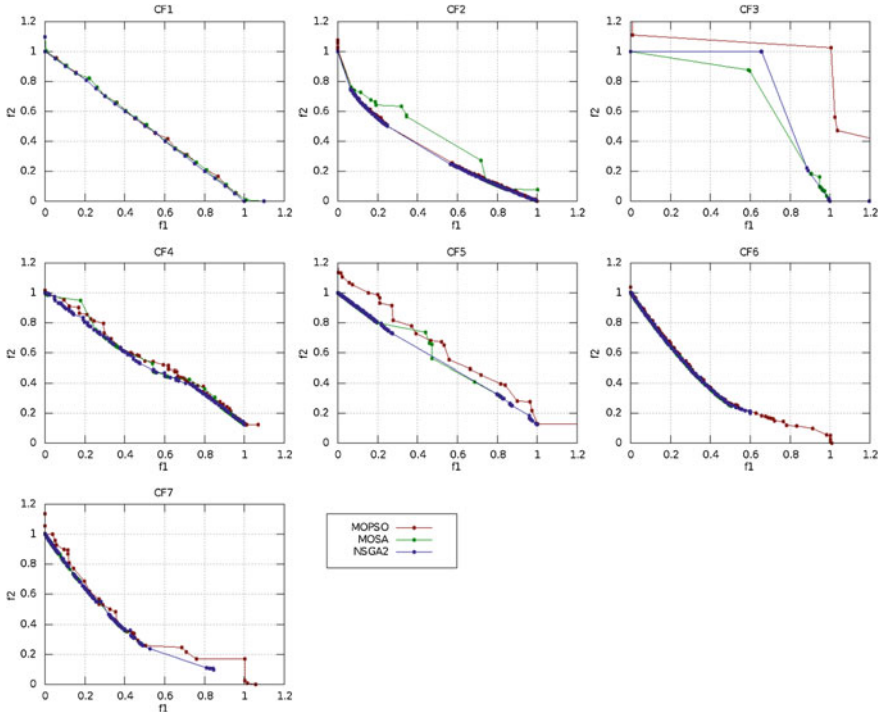

Fig. 5.2 Pareto fronts for problems with 4 variables: CF1–CF7

5.3 Evaluation of the Multi-kernel Methods

To experiment the multi-kernel methods and evaluate the performance of the merge operator in the parallel multi-algorithm kernel, two combinations are considered: first with two NSGAII kernels, and secondly with a MOSA and a NSGAII. The resultant POF is shown in Figs. 5.5 and 5.6, the configurations used are described in Table 5.1.

From the results it can be concluded that choosing the best elements of each kernel is the best approach. Also, it can be observed that the best between shuffle and sort by objective is dependent on the problem, where in CF1, CF2, and CF3 the shuffle method is better and in CF5, CF6, and CF7 the sort by objective tends to be better.

To evaluate the performance of the multi-kernel algorithms four combinations of two kernels were tested. The configurations used in the test are described in Table 5.2.

In Fig. 5.6 the results obtained for the four multi-kernel combinations are presented.

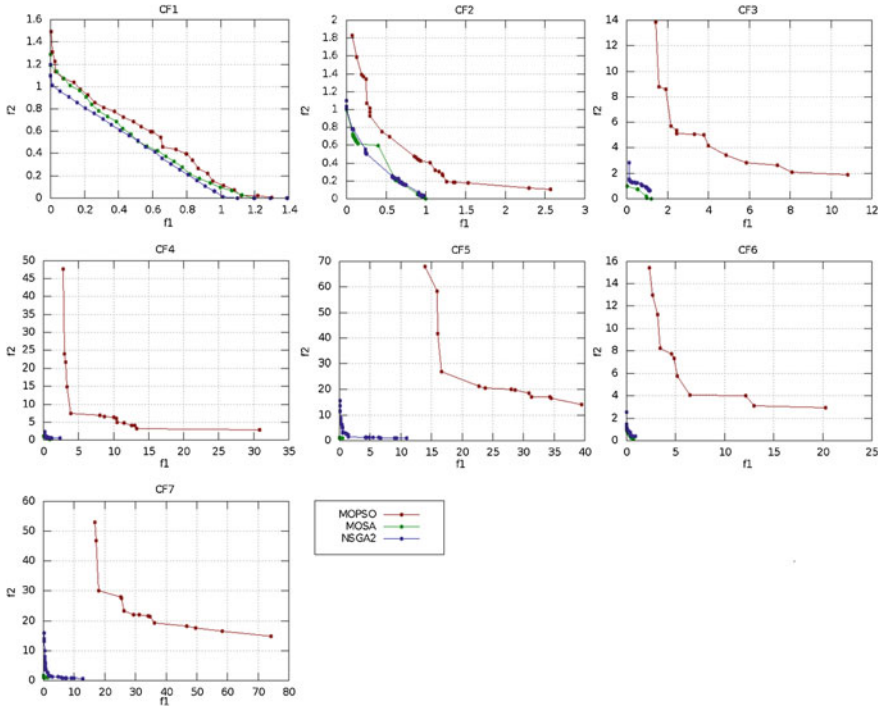

Fig. 5.3 Pareto fronts for problems with 30 variables: CF1–CF7

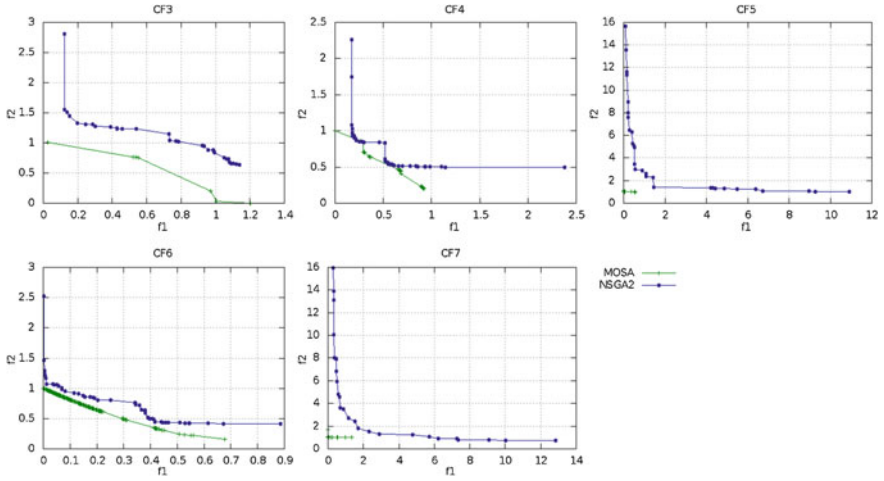

Fig. 5.4 Detail Pareto fronts for CF3–CF7 problems with 30 variables (MOSA and NSGAII)

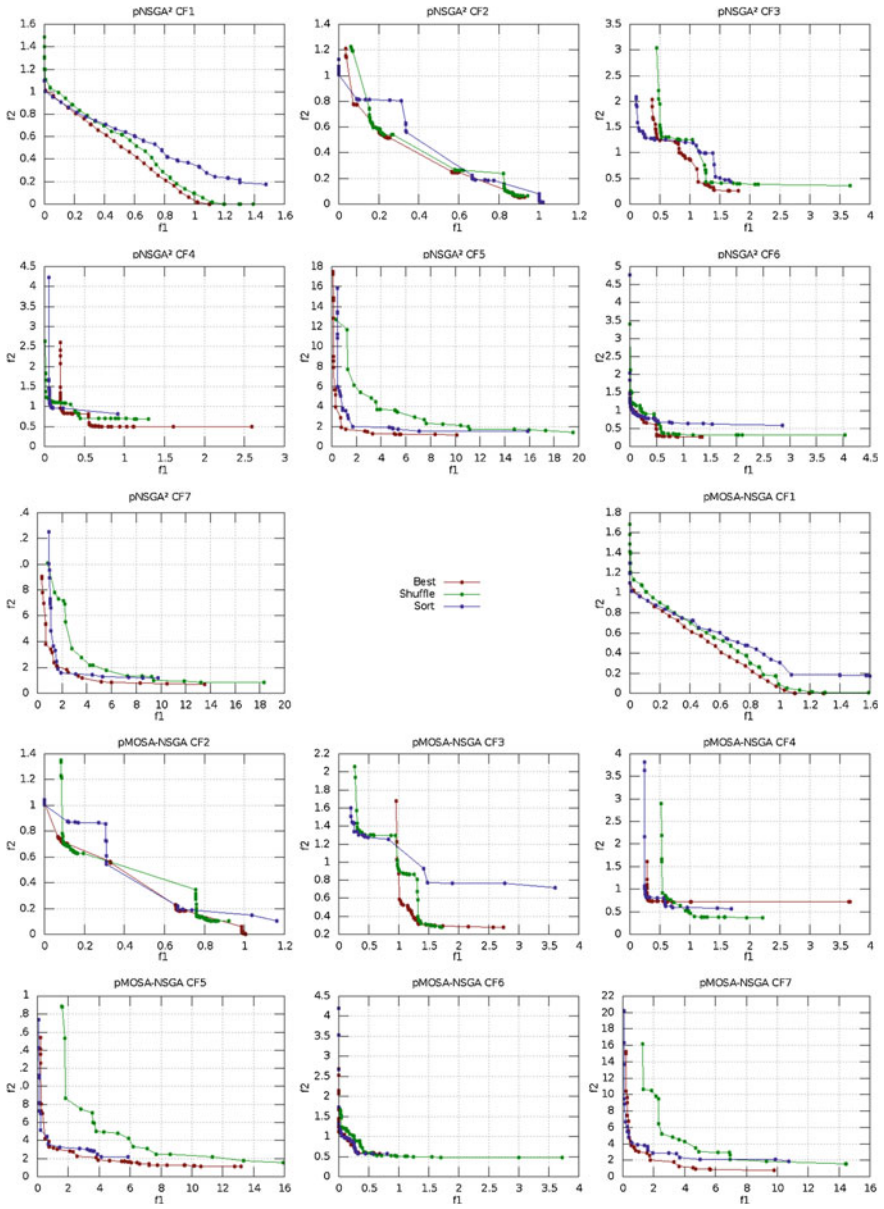

Fig. 5.5 Two parallel NSGA-II and parallel MOSA and NSGA-II with the three merge strategies: CF1–CF7

On the parallel execution of kernels, two ideas were experimented, one is to segment the space focusing the search done by the different kernels in different areas of the search space, and the other is to use different methods to explore the

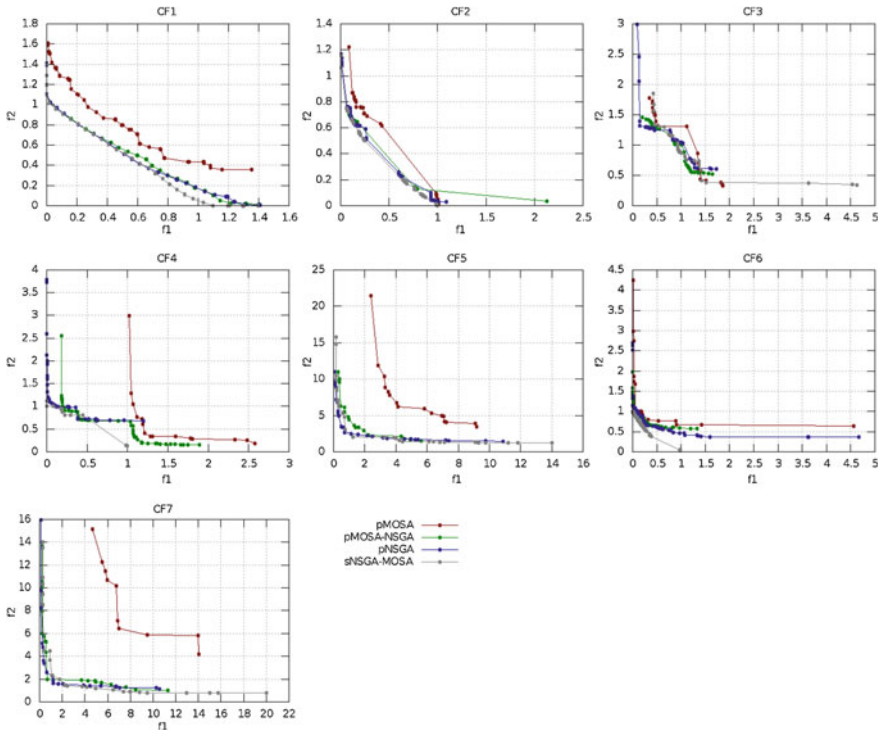

Fig. 5.6 Hybrid Pareto fronts for problems with 30 variables

Table 5.1 Parallel execution configurations

	Kernel1	Kernel2	Merge	Strategy
pNSGA^2	NSGAII(64,2300)	NSGAII(64,2300)	Step%30	Split sorted
				Split shuffle
				Select best
pMOSA-NSGAII	MOSA(64,2300)	NSGAII(64,2300)		Split sorted
				Split shuffle
				Select best

same space. The first idea leads to the pMOSA^2 and pNASGII^2 combinations, while the second to the pMOSA-NSGAII. Having a GA followed by SA is common in the literature due to the global nature of the GA that favors the exploration of the search space, and the local nature of the SA that favors the exploitation of the global solutions, hence the sequential sNSGAII-MOSA combination.

One interesting and unexpected fact when mixing the kernels was that mixing two MOSA, which delivers the best Pareto fronts for many problems when using a single kernel, delivers the worst Pareto for all the problems in the multi-kernel case.

Table 5.2 Hybrid configuration parameters

	Kernel1	Kernel2	Merge	Strategy
pMOSA^2	MOSA(64,2300)	MOSA(64,2300)	Step%300	Parallel-split sorted
pNSGA^2	NSGAII(64,2300)	NSGAII (64,2300)	Step%300	Parallel-split sorted
pMOSA-NSGAII	MOSA(64,2300)	NSGAII (64,2300)	Step%300	Parallel select best
sNSGAII-MOSA	NSGAII (128,2000)	MOSA(128,300)	Step% 2000	Seq.–all elements

Another point is that NSGAII produces good results when mixed with either another NSGAII or with MOSA. From the four tested combinations the best results are obtained with the sequential combination of NSGAII followed by MOSA. This result could be explained by the efficient exploration of the search space performed by the NSGAII and then applying the MOSA local search to further optimize the already "good" global solution.

These tests were made to confirm the performance of the algorithms that were implemented as made to prove the potential of the hybrid multi-objective multi-kernel methods; further study is required for each kernel individually, and also, in combinations of them. The test was also crucial to identify that the results of the combination of kernels cannot be easily foreseen, as illustrated by the case with pMOSA^2

5.4 Conclusions

In this chapter, mathematical problem optimizations were presented, some examples of combinations are introduced showing that mixing technics is nontrivial. As shown in the case of MOSA, albeit being the best in solo, the combination of two MOSA in parallel gives the worst result for the experimented multi-kernel combinations of algorithms.

It was also confirmed that the use of one algorithm with good global search properties, to initial explore the search space, and then use another one, with better local behavior, to best tune the solutions is a possible and viable approach. These tests should be considered a starting point for the future research on the combinations of optimization strategies applied to analog circuit.

Reference

1. Zhang, Q., Zhou, A., Zhao, S., Suganthan, P.N., Liu, W., Tiwari, S.: Multiobjective optimization Test Instances for the CEC 2009 Special Session and Competition (2009)

Chapter 6
Results for Analog IC Design

Abstract This chapter presents and discusses the results obtained with the implemented kernels in the sizing optimization of two analog differential amplifier circuits, a single-stage differential amplifier that uses voltage combiners, and a two-stage miller amplifier. The chapter starts with a brief exposition of the circuits and corresponding optimization problem, then the performance of the single-kernel methods is compared. Finally, the multi-kernel combination sMOSA-NSGAII is compared with the single-kernel methods, showing the potential of hybrid solutions.

Keywords Analog IC design · Automatic circuit sizing · Circuit optimization · Electronic design automation · Computer-aided-design

6.1 Differential Amplifier Circuit Problem

Before moving to the circuit optimizations, a brief preliminary note on the amplifier circuits considered is given. A common tradeoff in the design of amplifiers is between current (power) consumption and bandwidth (speed).

Traditionally, these conflicting objectives are reflected in the Figure-of-Merit (FOM) shown in (6.1), where GBW is the gain-bandwidth product, C_{load} is the load capacity, and I_{DD} is the current consumption. By maximizing the FOM, the power consumption is minimized and the bandwidth product is maximized. This FOM is commonly used by designers to access the energy efficiency of the achieved solution.

$$\text{FOM} = \frac{\text{GBW} \times C_{\text{load}}}{I_{\text{DD}}} \left[\frac{\text{MHz} \times \text{pF}}{\text{mA}}\right] \tag{6.1}$$

The measures of the circuit's performance are done using a test bench and circuit simulator. A test bench is a circuit that is used only in simulation, i.e., it is not

R. Lourenço et al., *AIDA-CMK: Multi-Algorithm Optimization Kernel Applied to Analog IC Sizing*, SpringerBriefs in Computational Intelligence,
DOI 10.1007/978-3-319-15955-3_6

intended to be fabricated. It is used to simulate, load, and provide the means to measure the performance figures of the circuit under test.

Two analog differential amplifier circuits, a single-stage differential amplifier that uses voltage combiners to boost the gain, and a two-stage miller amplifier are used to evaluate the performance of the various optimization methods. In the case of the single-stage amplifier, a typical 6 pF all capacitive load was considered, while in the two-stage miller amplifier was loaded with a 10 M resistor in parallel with 1 pF capacitor and biased with a current of 10 μA.

The performance figures of both circuits are measured with Mentor Graphics' Eldo® circuit simulator using operating point analysis and small-signal response analysis (AC). The extracted measures can then be used to define objectives and constraints.

The following subsections describe the circuits in detail and define the optimization problem that will be handled by the multiple methods.

6.1.1 Single-Stage Amplifier with Gain Enhancement Using Voltage Combiners

The first amplifier circuit is the single-stage amplifier topology using voltage combiners proposed in [1], whose circuit schematic is shown in Fig. 6.1.

The device sizes, which are listed in Table 6.1, constitute the variables in the optimization process, and their ranges define the search space that is explored during the sizing procedure. The variable names and ranges, considering both maximum and minimum values, as well as the precision of the search grid are presented in Table 6.1.

The optimization variables are the following device model parameters: finger widths, lengths, and number of fingers of the transistors. The index number in each

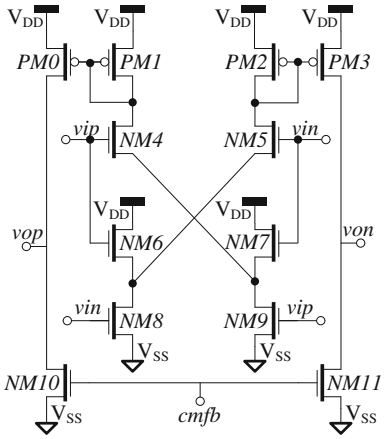

Fig. 6.1 Single-stage gain enhanced amplifier schematic

Table 6.1 Single-stage gain enhanced amplifier optimization variables and ranges

Variable (unit)	Minimum	Grid unit	Maximum
l0, l1, l4, l6, l8, l10 (nm)	120	10	1,000
w0 w1 w4 w6 w8 w10 (μm)	1	0.1	10
nf0, nf1, nf4, nf6, nf8, nf10	1	2	8

variable is according to the device names in Fig. 6.1. Note that the devices are paired: PM0 is equal to PM3; PM1 is equal to PM2; NM4 is equal to NM5; NM6 is equal to NM7; NM8 is equal to NM9; and NM10 is equal to NM11. Also, all variable ranges respect the technology available limits, in order to provide physically implementable solutions.

As stated before, the design objectives are the minimization of current consumption and maximization of bandwidth, and the circuit specifications considered in all optimization processes are presented in Table 6.2.

The criteria used to set the constraints were the following: all devices are set to operate in the moderate/strong inversion region, therefore, their overdrive voltage is required to be greater than or equal to 100 mV, while all devices are to be working in saturation, hence their saturation margin is required to be greater than or equal to 50 mV. To ensure stability the phase margin must be larger than 60°, and a 50 dB gain must be ensured.

Applying the procedure described in Chap. 3 leads to the multi-objective optimization problem formulation where the two objectives to be minimized are shown in (6.2), the constraints are shown in (6.3), and the search space is $[120, 130, \ldots, 1000]^6 \times [1.0, 1.1, \ldots, 10]^6 \times [1, 2, \ldots, 8]^6$.

$$\begin{aligned} f_1(x) &= \text{IDD} \\ f_2(x) &= -\,\text{GBW} \end{aligned} \tag{6.2}$$

Table 6.2 Gain enhanced amplifier optimization constraints

Circuit performance	Specification
Figure-of-merit (FOM)	≥1,000 MHz pF/mA
Current consumption (IDD)	≤350 μA
Low-frequency gain (GDC)	≥50 dB
Gain bandwidth product (GBW)@6 pF	≥30 MHz
Phase Margin (PM)	≥60º
Overdrive (Vth–VGS) of the *n*th PMOS devices (OVP^n)	≥100 mV
Overdrive (VGS–Vth) of the NMOS devices (OVN^n)	≥100 mV
Saturation margin (V_{DSat}–V_{DS}) of the PMOS devices (DP^n)	≥50 mV
Saturation margin (V_{DS}–V_{DSat}) of the NMOS devices (DN^n)	≥50 mV

$$
\begin{aligned}
g_1(x) &= \frac{\text{FOM} - 1000\,\text{MHz pF/mA}}{|-\ 1000\,\text{MHz pF/mA}|} \\
g_2(x) &= \frac{350\,\mu A - \text{IDD}}{|350\,\mu A|},\ g_3(x) = \frac{\text{GDC} - 50\,\text{dB}}{|50\,\text{dB}|} \\
g_4(x) &= \frac{\text{GBW} - 30\,\text{MHz}}{|30\,\text{MHz}|},\ g_5(x) = \frac{\text{PM} - 60^\circ}{|60^\circ|} \\
g_{6+d}(x) &= \frac{\text{OVP}^d - 100\,\text{mV}}{|100\,\text{mV}|},\ g_{10+d}(x) = \frac{\text{DP}^d - 50\,\text{mV}}{|50\,\text{mV}|} d = 0{:}3 \\
g_{10+d}(x) &= \frac{\text{OVN}^d - 100\,\text{mV}}{|100\,\text{mV}|},\ g_{22+d}(x) = \frac{\text{DN}^d - 50\,\text{mV}}{|50\,\text{mV}|} d = 4{:}11
\end{aligned}
\tag{6.3}
$$

6.1.2 Two-Stage Miller Amplifier

The other differential amplifier topology considered in this work is the two-stage miller operational amplifier, whose schematic is shown in Fig. 6.2.

Again, the optimization variables are the width, length, number of fingers and number of rows of the MOS devices, and the length and number of fingers of the MOM capacitor. The variable ranges are indicated in Table 6.3.

Like before, the circuit's performance figures are measured from the simulation results. Table 6.4 indicates the design specifications for this circuit working as a versatile low power DC buffer.

The previous design specifications lead to the optimization problem that is to be considered in the study of the multi-objective optimization of the two-stage amplifier where the two objectives to be minimized are again defined by (6.2), the constraints are now the ones defined in (6.4), and the search space is $[120,130,..,1000]^4 \times [1.0, 1.1, \ldots, 10]^4 \times [1, 2, \ldots, 20]^6 \times [4.4, 4.5, \ldots, 100] \times [14, 16, \ldots, 198]$.

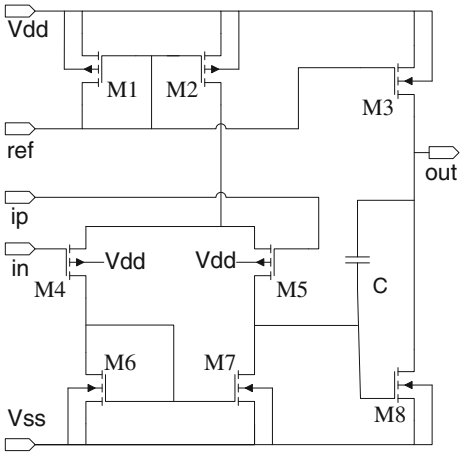

Fig. 6.2 Two-stage operational amplifier schematic

Table 6.3 Two-stage amplifier optimization variables and ranges

Variable (unit)	Min.	Grid unit	Max.
l1, l4, l6, l8 (nm)	120	5	1,000
w1 w4 w6 w8 (μm)	1	0.1	10
nf1, nf2, nf3, nf4, nf6, nf8	1	2	200
lc (μm)	4.4	0.1	100
nfc	14	2	198

Table 6.4 Two-stage amplifier specifications

Circuit performance	Constraint
Current consumption (IDD)	≤80 μA
Low-frequency gain (GDC)	≥50 dB
Unity gain frequency (GBW)	≥1 MHz
Phase margin (PM)	≥55°
Power supply rejection ratio (PSRR)	≥55 dB
Slew rate (SR)	≥0.8 V/μs
Offset voltage (V_{off})	≤1 mV
Noise RMS (No)	≤400 μV_{rms}
Noise density (Sn)	≤100 nV/$\sqrt{\text{Hz}}$
Overdrive (Vth–VGS) of the *n*th PMOS devices (OVP^n)	≥100 mV
Overdrive (VGS–Vth) of the NMOS devices (OVN^n)	≥100 mV
Saturation margin (V_{DSat}–V_{DS}) of the PMOS devices (DP^n)	≥50 mV
Saturation margin (V_{DS}–V_{DSat}) of the NMOS devices (DN^n)	≥50 mV

$$
\begin{aligned}
g_1(x) &= \frac{80\,\mu\text{A} - \text{IDD}}{|80\,\mu\text{A}|} \\
g_2(x) &= \frac{\text{GDC} - 50\,\text{dB}}{|50\,\text{dB}|},\ g_3(x) = \frac{\text{GBW} - 1\,\text{MHz}}{|1\,\text{MHz}|} \\
g_4(x) &= \frac{\text{PM} - 55^\circ}{|55^\circ|},\ g_5(x) = \frac{\text{PSRR} - 55\,\text{dB}}{|55\,\text{dB}|} \\
g_6(x) &= \frac{\text{SR} - 0.8\,\text{V}/\mu\text{s}}{|0.8\,\text{V}/\mu\text{s}|},\ g_7(x) = \frac{1\,\text{mV} - V_{\text{off}}}{|1\,\text{mV}|} \\
g_8(x) &= \frac{400\,\mu\,\text{V}_{\text{rms}} - \text{No}}{|400\,\mu\,\text{V}_{\text{rms}}|},\ g_9(x) = \frac{100\,\text{nV}/\sqrt{\text{Hz}} - \text{Sn}}{|100\,\text{nV}/\sqrt{\text{Hz}}|} \\
g_{9+d}(x) &= \frac{\text{OVP}^d - 100\,\text{mV}}{|100\,\text{mV}|},\ g_{17+d}(x) = \frac{\text{DP}^d - 50\,\text{mV}}{|50\,\text{mV}|}\,d = 1{:}5 \\
g_{9+d}(x) &= \frac{\text{OVN}^d - 100\,\text{mV}}{|100\,\text{mV}|},\ g_{17+d}(x) = \frac{\text{DN}^d - 50\,\text{mV}}{|50\,\text{mV}|}\,d = 6{:}8
\end{aligned}
\tag{6.4}
$$

6.2 Circuit Optimization Using the Single-Kernel Algorithms

To study the three single-kernel multi-objective strategies available in AIDA-CMK, a fixed number of evaluations (circuit simulations) were used to provide a fair ground for comparison of the optimization strategies. The total amount of simulations in the Single-Stage Amplifier optimization was 64,000. Two combinations of the number of elements and the number of iterations were considered, in Runset I these values were {64, 1,000}, respectively, and in Runset II the values were {128, 500}. For the two-stage amplifier the number of iterations was doubled to further understand the behavior of the methods with more evaluations, leading to Runset I with 64 elements and 2,000 iterations, and Runset II with 128 elements and 1,000 iterations. All runsets include ten independent executions, using a different seed in the random number generator for each run, taken from a common set of seeds, i.e., the same seed was used for the same run of the different algorithms and to understand the evolution of the best solutions with the number of evaluations, intermediary results were stored. One additional run was executed using only NSGA-II considering a population of 256 and 5,000 generations (1,280,000 evaluations) to find a better estimate of the true Pareto Optimal Front (POF), giving a reference to assess the previously obtained solutions' quality.

6.2.1 Comparison of the Single-Kernel Algorithms in the Single-Stage Amplifier Optimization

Figure 6.3 shows the Pareto fronts for Runset I at different stages of the optimization process, namely at 6,400, 12,800, 25,600, 38,400, 51,200 and 64,000 simulations, and Fig. 6.4 shows the same for Runset II.

The evolution of the best FOM, bandwidth and current consumption with the number of simulations for each optimization kernel in both runsets is illustrated in Fig. 6.5.

The analysis of the results shows that NSGA-II is much more efficient than MOPSO or MOSA, requiring fewer simulations to achieve the same solutions, and it is more consistent throughout the 10 runs. Unlike the results in [2], where even using MOSO and MOPSO, multiple oscillators are designed showing the state-of-the-art FOMs even though, as in this work, the FOM was not being explicitly optimized. In this work, NSGA-II clearly outperforms the other methods for the Amplifier designs which are more constrained problems, making the search for feasible solution harder. Besides the difference in the obtained solutions, it is also important to note that NSGA-II consistently get feasible solutions since very early in the optimization process, while the other methods struggle to find feasible solutions. Many of the MOPSO runs did not get any feasible solution in the 64k

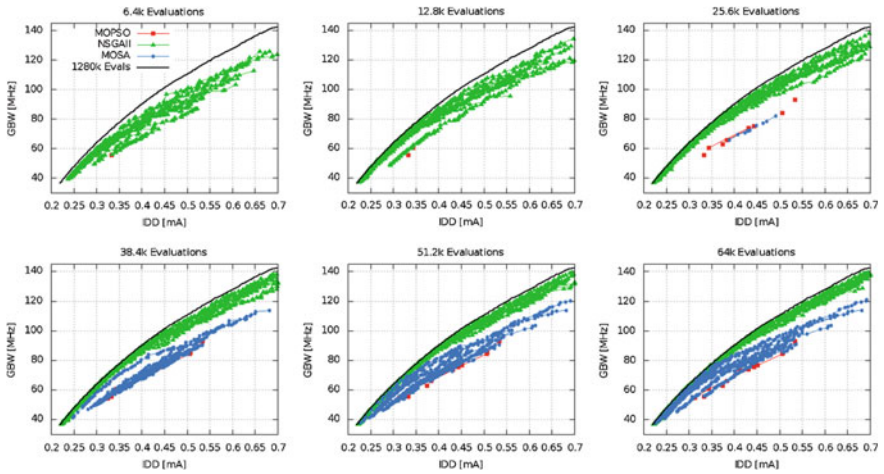

Fig. 6.3 Pareto fronts for the different runs of NSGA-II, MOPSO, and MOSA for Runset I

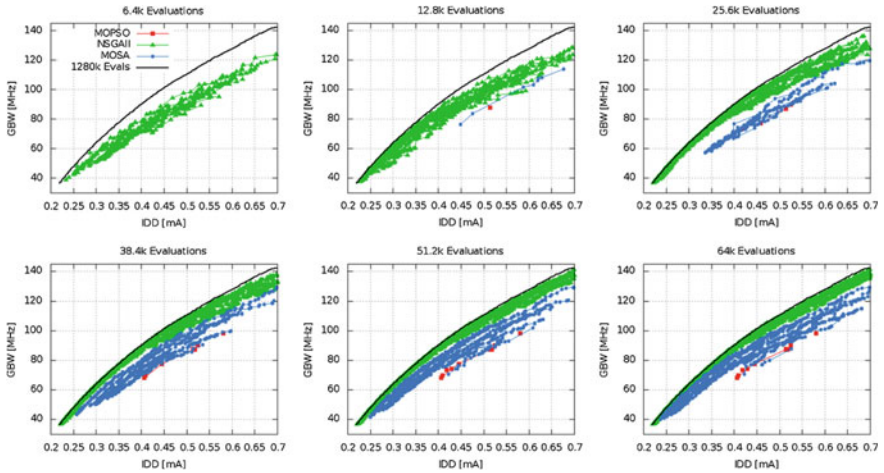

Fig. 6.4 Pareto fronts for the different runs of NSGA-II, MOPSO and MOSA for Runset II

evaluations, MOSA performed better, but still requiring a considerably larger number of evaluations when compared to NSGA-II.

Comparing the performance between runsets, NSGA-II behaves similarly in both cases, reaching close to the "true" POF, MOPSA struggles in both cases, and MOSA get closer to the "true" POF in Runset II shortening the gap in the high frequency side of the POF.

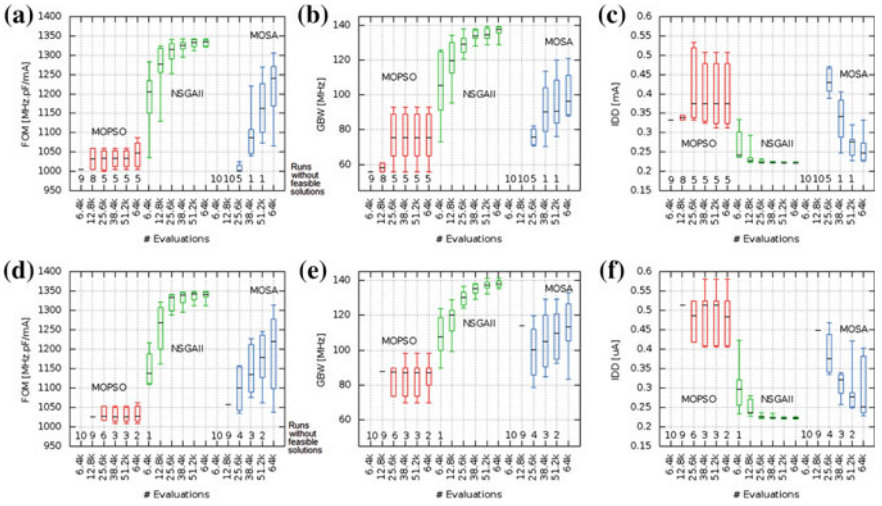

Fig. 6.5 Evolution of the best FOM, GBW, and IDD with the number of simulations

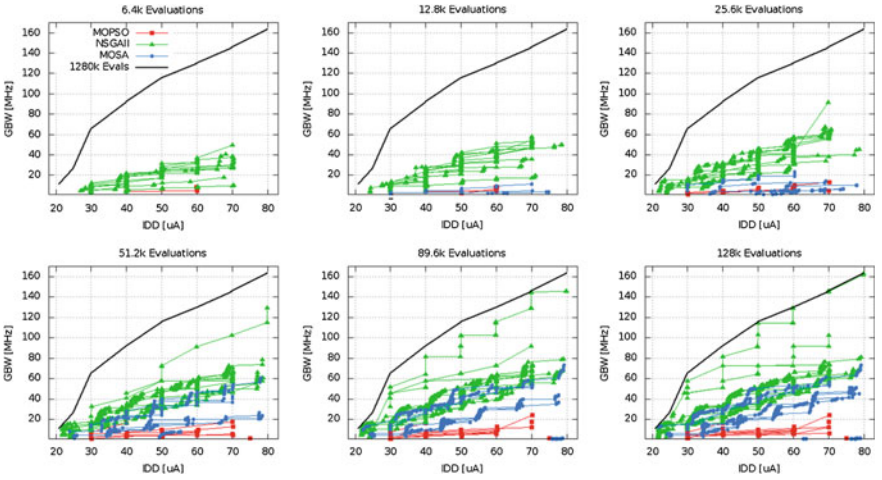

Fig. 6.6 Pareto fronts for the different runs of the NSGA-II, MOPSO and MOSA on the two-stage amplifier problem for Runset I

Note that, as can be seen in Fig. 6.7a where from 51.2 to 64k evaluations of the MOSA the worst FOM actually gets worse; this happens because the runs without feasible solutions are not accounted to derive the box plots, when one or more of the runs that did not have any feasible solutions before, now have a new feasible solution, wherein new solutions may worsen the worst performance.

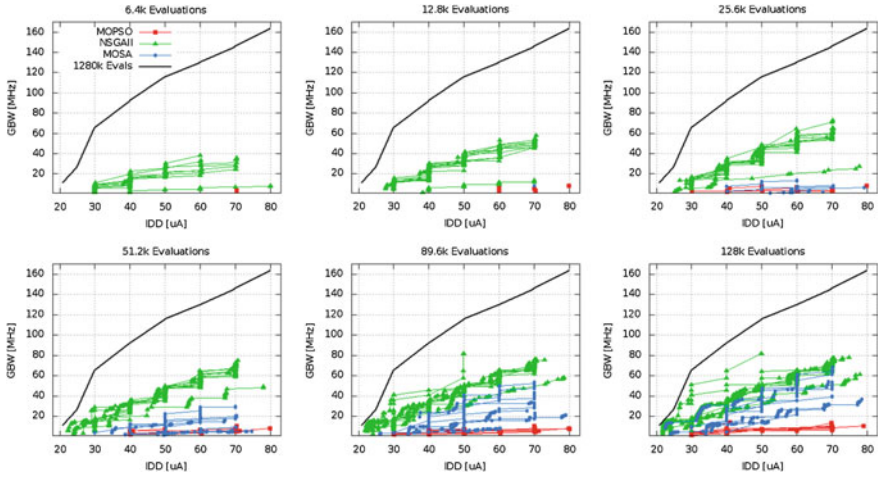

Fig. 6.7 Pareto fronts for the different runs of the NSGA-II, MOPSOm and MOSA on two-stage amplifier problem for Runset II

6.2.2 Comparison of the Single-Kernel Algorithms in the Two-Stage Amplifier Optimization

Figure 6.6 shows the Pareto fronts for Runset I at different stages of the optimization process, namely at 6,400, 12,800, 25,600, 51,200, 89,600, and 128,000 simulations, and Fig. 6.7 shows the same for Runset II.

The evolution of the best FOM, bandwidth, and current consumption with the number of simulations for each optimization kernel in both runsets is illustrated in Fig. 6.8.

The analysis of the results shows again that NSGA-II is more efficient than MOPSO or MOSA, attaining better results at the end and throughout the optimization process.

Unlike the previous circuit, where both NSGA-II and MOSA manage to almost reach the "true" POF with 64,000 simulations, in this circuit all algorithms stayed reasonably far from it despite the extra simulations considered in this test case, showing that different circuit/specifications present different results for similar algorithm configurations in similar search spaces.

Another relevant aspect is the fact that both MOSA and MOPSO struggle to achieve feasibility. However, although MOSA took longer to find feasible solutions (and in some runs it did not find any), when feasible solutions were found, it almost managed to catch NSGA-II.

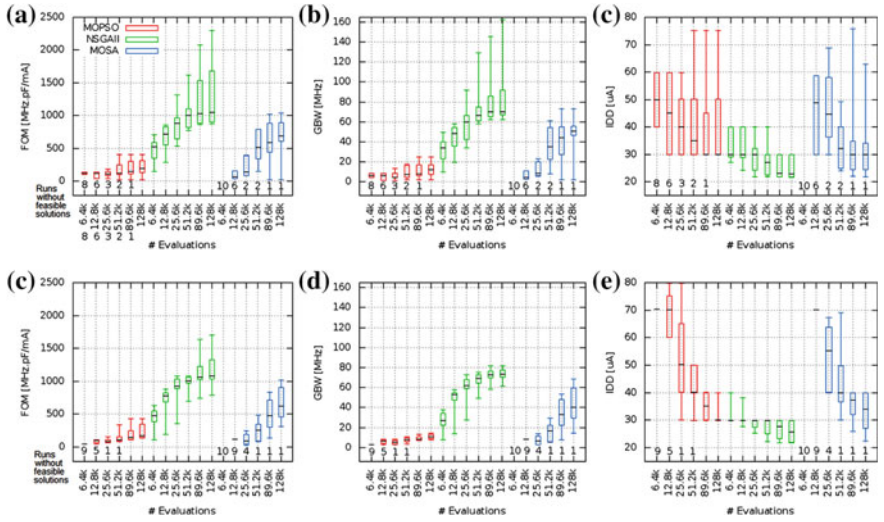

Fig. 6.8 Evolution of the best FOM, GBW, and IDD with the number of simulations in the two-stage amplifier runsets

In terms of comparing results of Runset I and Runset II, it is seen that the average results are almost the same. However, two points to note are: first, in Runset I some runs get closer to the true POF showing that more iterations with fewer elements is better to push the quality of the solutions, second also in Runset I it was more difficult to achieve feasibility (this is also true for the Single-Stage amplifier results) showing that fewer elements limit the exploration capabilities of the optimization methods.

The two-stage amplifier was also optimized to study the multi-objective optimization considering multiple kernel combinations, the hybrid configuration, sequential NSGA-II-MOSA (sNSGA-MOSA), was used. This choice was made based on the test results using mathematical functions. The total amount of simulations in each test run was 128.000, the number of elements was 128, and the number of iterations was 1,000. The sNSGA-MOSA configuration used again, switching the kernels on the 700th iteration, resulted in 700 NSGA iterations followed by 300 MOSA iterations. Figure 6.9 shows the Pareto fronts obtained from a couple of independent runs.

Again the sNSGA-MOSA and NSGA-II outperform the MOSA. The NSGA-II and sNSGA-MOSA presented similar results in the center of the POF, sNSGA-MOSA clearly outperforms the NSGA at the edges of the POF.

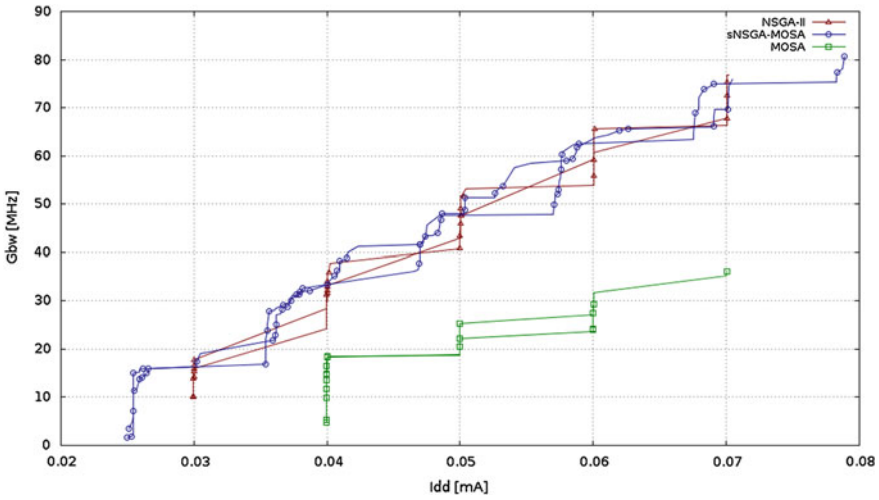

Fig. 6.9 Two-stage operational amplifier results

6.3 Conclusions

In this chapter the circuit problem optimizations are presented and it is shown that the optimization of analog circuit is very different from the optimization of the mathematical benchmark functions, considered in Chap. 5. Although defined similarly, the optimization of these two types of problems is not trivial, and the best strategy to optimize one is not necessarily the best for the other. Given the example of the MOSA, it performed best in solo on mathematical functions, but proved to be the worst option for the circuit problems. It was also shown that the use of one algorithm to initially explore the search space and then use another to further optimize until a good solution is found is a possible and viable approach, but further testing on the merge strategies of the algorithms must be conducted in order to tune the optimization and improve the quality of the results found.

References

1. Póvoa, R., Lourenço, N., Horta, N., Santos-Tavares, R., Goes, J.: Single-stage amplifiers with gain enhancement and improved energy-efficiency employing voltage-combiners. In: 21st IFIP/IEEE International Conference on Very Large Scale Integration, Istambul, 2013
2. Póvoa, R., Lourenço, R., Lourenço, N., Canelas, A., Martins, R., Horta, N.: Synthesis of LC-oscillators using rival multi-objective/multi-constraint optimization kernels. In: Performance Optimization Techniques in Analog, Mixed-Signal, and Radio-Frequency Circuit Design. IGI Global, 2014

Chapter 7
Conclusion and Future Work

Abstract This chapter presents the conclusions for this work, and the future directions for the continuous development of AIDA and the circuit optimization framework.

Keywords Analog IC design · Electronic design automation · AIDA framework

7.1 Conclusions

The work presented in this book corresponds to an innovative IC design automation approach by implementing an abstraction layer between the circuit optimization and the optimization engine. Moreover, by creating a flexible optimization framework that implements an abstraction layer, two important goals were achieved: first it was possible in a very short period to develop comparative studies of three multi-objective optimization methods, namely the MOSA, NSGAII, and MOPSO; secondly, two hybridization methods were proposed and tested in real analog circuits showing the potential of the implemented framework to help in the development and tailoring of innovative optimization methods for analog ICs.

Finally, the proposed objectives for this work were achieved and a new optimizer was created.

7.2 Further Work

In analog design automation, the development of new and better approaches is always necessary. There is still a long way to go in this domain; the improvement on productivity of analog design is an economic demand, from this work, and in its application to analog design there are some suggestions for future research which may improve the development of new and better automation tools. The first and

R. Lourenço et al., *AIDA-CMK: Multi-Algorithm Optimization Kernel Applied to Analog IC Sizing*, SpringerBriefs in Computational Intelligence,
DOI 10.1007/978-3-319-15955-3_7

obvious suggestion is to reap the benefits of the implemented framework by experimenting new methods and approaches to the analog circuit sizing and optimization.

Another suggestion is the creation of a public circuit optimization benchmark that is well-defined and does not depend on confidential technology information that lay the building blocks for an open access common ground enabling to compare the implemented methods between research groups. While this clearly would be an incredibly valuable tool in the circuit sizing and optimization domain and although conceptually simple, the nature of analog design and the amount of trade secrets and confidential implementation details make it extremely difficult to legally distribute a realistic collection of analog circuits that can be optimized by anyone anywhere in the world.

Finally, and although simple, unity testing was implemented for many of the developed classes, integration and functional testing with automatic periodic regressions while extremely valuable for the maintenance of the framework, were out of the scope of this work. Creating such an infrastructure, while not necessarily scientifically relevant, would further ease the development of new and innovative methods for analog IC optimization.