



INAOE

Optimization of Analog Circuits by Applying Evolutionary Algorithms

by

Ivick Guerra Gómez

A Dissertation

Submitted to the Program in Electronics Science,
Electronics Science Department
in partial fulfillment of the requirements for the degree of

PH.D. IN ELECTRONICS SCIENCE

at the

National Institute for Astrophysics, Optics and Electronics
July 2012
Tonantzintla, Puebla

Advisors:

Dr. Esteban Tlelo Cuautle, INAOE.
Dr. Trent McConaghy, Solido Design Automation Inc.

©INAOE 2012

All rights reserved

The author hereby grants to INAOE permission to reproduce and
to distribute copies of this thesis document in whole or in part



To Leonor, Dafne and Sofía

Abstract

Unlike automated digital design, analog circuit designers require experience to develop skills, and to avoid spending a lot of time understanding all the aspects involved around a specific design such as nonlinearities, parasitics, performances, trade-offs, etc.

The continuing size reduction of electronic devices along with their shorter life cycle imposes design challenges to discover or to optimize the performances of modern electronic systems; such as: wireless services, telecom, mobile computing and media applications. Fortunately, those design challenges can be overcome through the development of Electronic Design Automation (EDA) tools.

In the analog domain, circuit optimization tools have demonstrated their usefulness in addressing design problems taking into account downscaling technological aspects. However, those EDA tools still have the lack of taking into account some design constraints when applied to a multiple objective design problem.

On the one hand, Evolutionary Algorithms (EAs) have demonstrated their suitability in solving nonlinear multi-objective design problems with multiple constraints, providing a set of feasible design solutions from which several insights and trade-offs among the circuit performance objectives can be deduced. On the other hand, still the application of EAs in optimizing the biases and sizes of analog circuits, have hard shortcomings to be improved, for instance: guarantee of convergence, run-time and incorporation of variation aware techniques.

This Thesis is focused on the application of multi-objective EAs in the optimization of analog circuits including nanometer technology. The EAs have been programmed to work with different genetic operators over different kinds of circuit objectives, variables and design technologies. A major contribution is presented by introducing an automatic current-bias distribu-

tion to accelerate convergence of the EAs in optimizing analog circuits. Another contribution is introduced through the implementation of a procedure oriented to compute process variations while diminishing the number of simulations. That is, optimal solutions are found by adapting a simulation budget allocation procedure to efficiently distribute the number of simulations. Finally, this Thesis includes an appendix to describe an EDA tool based on EAs for biasing and sizing analog circuits and taking into account process variations issues.

Resumen

A diferencia del diseño digital automatizado, los diseñadores de circuitos analógicos requieren experiencia para desarrollar habilidades y así evitar desperdiciar tiempo en entender todos los aspectos que están involucrados alrededor de un diseño específico, como es el caso de las no-linealidades, efectos parásitos, rendimiento del circuito, compromisos de diseño, etc.

La continua reducción del tamaño de los dispositivos electrónicos aunado a su ciclo de vida cada vez menor, imponen retos de diseño para descubrir u optimizar el rendimiento de los sistemas electrónicos modernos. Tal es el caso de sistemas inalámbricos, telecomunicaciones, dispositivos portátiles y aplicaciones multimedia. Afortunadamente, dichos retos de diseño pueden superarse a través del desarrollo de herramientas de Diseño Electrónico Automatizado (EDA).

En el dominio analógico, las herramientas para la optimización de circuitos han demostrado una gran utilidad para abordar los problemas de diseño tomando en cuenta los aspectos de escalamiento tecnológico. Sin embargo, estas herramientas de EDA aún presentan deficiencias al tomar en cuenta algunos compromisos de diseño cuando se aplican a un problema de diseño multi-objetivo.

Por una parte, los Algoritmos Evolutivos (EAs) han demostrado ser idóneos para resolver problemas de diseño no lineales multi-objetivo con múltiples compromisos, aportando un conjunto de soluciones factibles. A partir de este conjunto de soluciones se pueden hacer algunas conjeturas para establecer relaciones entre los diferentes rendimientos de un circuito. Por otra parte, la aplicación de EAs en la optimización a base del dimensionamiento y polarización de circuitos analógicos, aún tiene muchos inconvenientes que deben ser superados, por ejemplo: garantizar la convergencia, el tiempo de cómputo y la incorporación de técnicas que tomen en

cuenta las variaciones de proceso.

Esta Tesis esta enfocada a la aplicación de EAs multi-objetivo en la optimización de circuitos analógicos de tecnologías nanométricas. Los EAs han sido programados para trabajar con diferentes operadores genéticos , diferentes tipos de objetivos de circuito, varias variables y distintas tecnologías de diseño. Se presenta una contribución al introducir una distribución de corrientes de polarización automáticamente para acelerar la convergencia de los EAs en la optimización de circuitos analógicos. También se presenta otra contribución a través de la implementación de un procedimiento para calcular las variaciones de proceso con un número menor de simulaciones.

Finalmente, esta Tesis incluye un apéndice donde se describe una herramienta EDA basada en EAs para la polarización y dimensionamiento de circuitos analógicos tomando en cuenta las variaciones de proceso.

Publications

Papers

- I. Guerra-Gómez, E. Tlelo-Cuautle, T. McConaghy, and G. Gielen, “Decomposition-Based Multi-Objective Optimization of Second-Generation Current Conveyors,” *IEEE International Midwest Symposium on Circuits and Systems*, pp. 220–223, August 2009.
- I. Guerra-Gómez, E. Tlelo-Cuautle, G. Reyes-Salgado, and Luis G. de la Fraga, “Non-sorting genetic algorithm in the optimization of unity-gain cells,” *International Conference on Electrical Engineering, Computing Science and Automatic Control*, pp. 445–450, Toluca-Mexico, November 2009.
- I. Guerra-Gómez, E. Tlelo-Cuautle, T. McConaghy, and G. Gielen, “Optimizing Current Conveyors by Evolutionary Algorithms Including Differential Evolution,” *International Conference on Electronics Circuits and Systems*, Special Session, pp. 259–262, Hammamet-Tunisia, December 2009.
- E. Tlelo-Cuautle, I. Guerra-Gómez, M. A. Duarte-Villaseñor, Luis G. de la Fraga, G. Flores-Becerra, G. Reyes-Salgado, C.A. Reyes-García and G. Reyes-Gómez, “Applications of Evolutionary Algorithms in the Design Automation of Analog Integrated Circuits,” *Journal of Applied Sciences*, vol. 10, no. 17, pp. 1859–1872, 2010.
- I. Guerra-Gómez, E. Tlelo-Cuautle, T. McConaghy, Luis G. de la Fraga and G. Gielen, “Sizing mixed-mode circuits by multi-objective evolutionary algorithms,” *IEEE International Midwest Symposium on Circuits and Systems*, pp. 813–816, August 2010.

- I. Guerra-Gómez, E. Tlelo-Cuautle and Luis G. de la Fraga, “Sensitivity analysis in the Optimal Sizing of Analog Circuits by Evolutive Algorithms,” *International Conference on Electrical Engineering, Computing Science and Automatic Control*, pp. 381–385, Chiapas-Mexico, September 2010.
- A. Sallem, I. Guerra-Gómez, M. Fakhfakh, M. Loulou, E. Tlelo-Cuautle, “Simulation-Based Optimization of CCIIs Performances in Weak Inversion,” *International Conference on Electronics Circuits and Systems*, pp. 661–664, Athens-Greece, December 2010.
- S. Polanco-Martagón, G. Reyes-Salgado, G. Flores-Becerra, I. Guerra-Gómez, E. Tlelo-Cuautle, L. Gerardo de la Fraga, M.A. Duarte-Villaseñor, Selection of MOSFET Sizes by Fuzzy Sets Intersection in the Feasible Solutions Space, *Journal of Applied Research and Technology*, vol. 10, no. 3, pp. 472-483, 2012.

Book Chapters.

- E. Tlelo-Cuautle, I. Guerra-Gómez, C. A. Reyes-García, and M. A. Duarte-Villaseñor, *Intelligent Systems for Automated Learning and Adaptation: Emerging Trends and Applications*, ch. 8: Synthesis of Analog Circuits by Genetic Algorithms and their Optimization by Particle Swarm Optimization, IGI Global, pp. 173–192, 2010.
- E. Tlelo-Cuautle, I. Guerra-Gómez, Luis G. de la Fraga, G. Flores-Becerra, S. Polanco Martagón, M. Fakhfakh, C.A. Reyes-García , G. Reyes-Gómez and G. Reyes-Salgado, *Intelligent Computational Optimization in Engineering* ch. Evolutionary Algorithms in the Optimal Sizing of Analog Circuits, Springer, pp. 110-138, 2011.
- I. Guerra-Gómez, M.A. Duarte-Villaseñor, E. Tlelo-Cuautle, Luis G. de la Fraga, Analysis, Design and Optimization of Active Devices, in *Integrated Circuits for Analog Signal Processing*, ISBN: 978-1-4614-1382-0, Springer, June 2012.

Acknowledgements

This work is dedicated to my family whom have supported me always. Also, my most sincere gratitude for the guidance of my advisors, Dr. Esteban Tlelo Cuautle and Trent McConaghy.

I want also thank Dr. Francisco Fernández, Dr. Gerardo de la Fraga, Dr. Gustavo Rodríguez, Dr. Guillermo Espinosa and Dr. Arturo Sarmiento for their advices and taking part of my thesis work.

Finally is my appreciation to the Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE) and the Consejo Nacional de Ciencia y Tecnología (CONACyT) for its support through the scholarship 27516/204240 and the projects 131839-Y and 48396-Y.

Contents

Abstract	i
Resumen	iii
Publications	v
Acknowledgements	vii
1 Introduction	1
1.1 Analog Circuit Optimization Tools: Categories and Classifications	2
1.2 EDA tools	4
1.3 Justification	8
1.4 Objectives	10
1.5 Thesis Organization	12
2 Evolutionary Algorithms	15
2.1 Introduction	15
2.2 Evolutionary Algorithms concepts	15
2.2.1 Individuals, Population, Evolutionary Operators and Objective Function	15
2.2.2 The General Evolutionary Algorithm	17
2.3 Multiobjective Optimization and Pareto Dominance	18
2.3.1 Multiobjective Design Problem	18
2.3.2 Pareto Dominance	19
2.3.3 Diversity and Efficiency	20

2.4	Genetic Operators	21
2.4.1	Crossover and Mutation	21
2.4.2	Differential Evolution (DE)	22
2.4.3	Simulated Binary Cross-Over Operator (SBX)	22
2.4.4	Polynomial Mutation	23
2.5	NSGA-II, MOEAD and MOPSO	24
2.5.1	Non-Dominated Sorting Genetic Algorithm II (NSGA-II)	24
2.5.2	Multi-Objective Evolutionary Algorithm based on Decomposition (MOEAD)	27
2.5.3	Multi-Objective Particle Swarm Optimization (MOPSO)	29
2.5.4	Behavior of NSGA-II, MOEAD and MOPSO on test functions ZDT	31
2.6	Summary	33
3	Circuit Optimization	37
3.1	Introduction	37
3.2	Optimization Methodology Framework	37
3.3	Circuit Optimization with SBX and DE	39
3.3.1	Multi-Objective Optimization Problem Formulation	39
3.3.2	Results	41
3.4	Circuit Optimization with NSGA-II, MOEAD and MOPSO	47
3.4.1	Multi-Objective Optimization Problem Formulation	49
3.4.2	Results	52
3.5	Summary	57
4	Automatic current-bias distribution	59
4.1	Introduction	59
4.2	Bias assignments in CMOS analog circuits by graph manipulations	59
4.3	Modeling the Transistor for Current Bias Partitioning	62
4.3.1	Modeling the Transistor	62
4.3.2	Current Bias Distribution	63
4.4	Circuits and graphs	64

<i>CONTENTS</i>	xi
4.4.1 Incidence Matrix	64
4.4.2 Depth First Search for biasing	65
4.5 Proposed Current-Branche s-Bias Assignment (CBBA) Approach	67
4.6 Application Examples	70
4.6.1 FC OTA Results	72
4.6.2 RFC OTA Results	81
4.7 Summary	84
5 Circuit Variation Analysis	89
5.1 Introduction	89
5.2 Variation Analysis for Analog Circuits	89
5.3 Sensitivity Optimization of Analog Circuits	93
5.3.1 Multi-Parameter Sensitivity Analysis	94
5.3.2 Proposed Optimization System Including Multi-Parameter Sensitivity Analysis	99
5.3.3 Example	101
5.4 OCBA in the Yield Circuit Optimization	107
5.4.1 Optimal Computing Budget Allocation (OCBA)	108
5.4.2 Proposed Optimization System Including Yield Analysis by Using OCBA	110
5.4.3 Application Example	112
5.5 Summary	117
6 Conclusions	119
A Circuit Optimizer Software Tool	123
A.1 Software input sections	124
A.1.1 HSPICE Section	125
A.1.2 Variables Section	126
A.1.3 Objectives Section	127
A.1.4 Constraint Section	128
A.1.5 Optimization parameters section	130

A.1.6	CBBA and Yield section	131
A.2	Buttons section	133
A.3	Software output	133
B	Transistor Models	135
B.1	Transistor Model for 0.35 μm technology	135
B.2	Transistor Model for 180 nm technology	139
B.3	Transistor Model for 90 nm technology	141

List of Figures

1.1	Performances Estimations	4
2.1	A chromosome example	16
2.2	Population example.	16
2.3	Pareto optimal set example for two objective functions.	19
2.4	Single-point crossover example for $n = 5$	21
2.5	Mutation example for the third gene.	22
2.6	Fast Nondominated Sort Algorithm example	26
2.7	ZDT functions with 5 variables for NSGA-II _{SBX}	32
2.8	ZDT functions with 5 variables for NSGA-II _{DE}	33
2.9	ZDT functions with 5 variables for MOEAD _{SBX}	34
2.10	ZDT functions with 5 variables for MOEAD _{DE}	35
2.11	ZDT functions with 5 variables for MOPSO.	35
3.1	Optimization Methodology Framework	38
3.2	Tested Circuits.	40
3.3	CCII ⁺ Voltage Optimization.	42
3.4	CCII ⁺ Current Optimization.	43
3.5	Three different Voltage Followers.	47
3.6	CFOAs.	48
4.1	Norator, nullator and nullor	62
4.2	Modeling a MOSFET with a nullor.	63

4.3	Current behavior model of a MOSFET.	63
4.4	Examples of outgoing currents branches.	64
4.5	First two loops of Algorithm 12.	70
4.6	Limit search space assignment.	70
4.7	Folded Cascode OTA.	71
4.8	Recycled Folded Cascode OTA	71
4.9	Voltage references for Vbn1, Vbn2, Vbp1 and Vbp2.	72
4.10	Search Space Limits for the FC OTA.	74
4.11	Solutions for the FC OTA with and without CBBA for NSGA-II.	79
4.12	Solutions for the FC OTA with and without CBBA for MOEAD.	79
4.13	Solutions for the FC OTA with and without CBBA for MOPSO.	80
4.14	Search Space Limits for the RFC OTA.	82
4.15	Solutions for the RFC OTA with and without CBBA for NSGA-II.	83
4.16	Solutions for the RFC OTA with and without CBBA for MOEAD.	83
4.17	Solutions for the RFC OTA with and without CBBA for PSO.	87
5.1	Example of process variation.	90
5.2	Acceptability and Tolerance Regions.	90
5.3	Classification of Support Variation Analysis	91
5.4	Optimization of SRN by applying NSGA-II.	97
5.5	Sensitivity of SRN with respect to x_1	98
5.6	Multi-parameter sensitivity of SRN function for f_1 and f_2	98
5.7	Feasible solutions for SRN after applying multi-parameter sensitivity analysis.	99
5.8	Flow Diagram.	100
5.9	Recycled Folded Cascode OTA.	101
5.10	Behavior of Multi-Parameter Sensitivity vs Generations.	107
5.11	Flow Diagram for Optimization including OCBA.	111
5.12	Accumulated simulations for the FC OTA with and without OCBA for NSGA-II.	114
5.13	Accumulated simulations for the FC OTA with and without OCBA for MOEAD.	114
5.14	Accumulated simulations for the FC OTA with and without OCBA for MOPSO.	115

A.1	Software tool modules and inputs.	124
A.2	Software tool main window.	125
A.3	HSPICE section.	126
A.4	Variables section.	127
A.5	Add new variable.	127
A.6	Objectives section.	128
A.7	Add new objective.	128
A.8	Constraints section.	129
A.9	Add new constraint.	130
A.10	Load Sat/Cut Constraints.	130
A.11	Build Sat/Cut library for HSPICE.	131
A.12	General and particular parameters for each EA.	132
A.13	Build Sat/Cut library for HSPICE.	132

List of Tables

2.1	Test Functions	32
2.2	Coverage metric for each method for ZDT functions with 5 variables	36
3.1	Measurements Library Example	39
3.2	Variables encoding for the CCII's	40
3.3	CCII's Voltage Optimization Results	44
3.4	CCII's Current Optimization Results	45
3.5	Coverage Metric for CCII's Current and Voltage Optimization	46
3.6	CFOA _{AA} encoding	50
3.7	CFOA _{AB} encoding	50
3.8	CFOA _{AC} encoding	50
3.9	CFOA _{BA} encoding	50
3.10	CFOA _{BB} encoding	51
3.11	CFOA _{BC} encoding	51
3.12	CFOA _{CA} encoding	51
3.13	CFOA _{CB} encoding	51
3.14	CFOA _{CC} encoding	52
3.15	Results of optimization for CFOA _{AA} , CFOA _{AB} and CFOA _{AC}	54
3.16	Results of optimization for CFOA _{BA} , CFOA _{BB} and CFOA _{BC}	55
3.17	Results of optimization for CFOA _{CA} , CFOA _{CB} and CFOA _{CC}	56
4.1	Encoding for the FC OTA and RFC OTA.	72

4.2	Transistor sizes for voltage references.	73
4.3	Current-branches-bias assignments for the FC OTA.	74
4.4	Optimal solutions for the FC OTA with NSGA-II.	76
4.5	Optimal solutions for the FC OTA with MOEAD.	77
4.6	Optimal solutions for the FC OTA with MOPSO.	78
4.7	Current-branches-bias assignments to the RFC OTA.	81
4.8	Optimal solutions for the RFC OTA with NSGA-II.	84
4.9	Optimal solutions for the RFC OTA with MOEAD.	85
4.10	Optimal solutions for the RFC OTA with MOPSO.	86
5.1	Encoding for the RFC OTA shown in Figure 5.9.	102
5.2	Best points for the RFC OTA without sensitivity analysis.	104
5.3	Best sizing solutions without sensitivity analysis for the RFC OTA.	104
5.4	Best points for the RFC OTA including sensitivity analysis.	106
5.5	Best sizing solutions including sensitivity analysis for the RFC OTA.	106
5.6	Optimal variation-aware solutions for the FC OTA.	116

List of Algorithms

1	General Pseudocode for an Evolutionary Algorithm	18
2	NSGA-II Algorithm	24
3	Fast Nondominated Sort	26
4	Crowding Distance Assignment	27
5	Build spread of N weight vectors ($M = 3$)	28
6	Build spread of N weight vectors for M objectives	28
7	MOEAD Algorithm	29
8	Pseudocode for MOPSO	30
9	Depth First Search Algorithm (dfs)	65
10	Depth First Search Algorithm Top-Down (dfs_{TD})	66
11	Distribution of Currents	68
12	Limit search space assignment procedure	69
13	Richardson Extrapolation	96
14	OCBA Pseudocode	109

Chapter 1

Introduction

Analog designers require experience to develop circuit design skills, spending a lot of time gathering experience to understand all the aspects involved around a specific design such as nonlinearities, parasitics, performances trade-offs, etc. Current electronics devices with shorter life cycle and the recent increase of portable electronic systems such as wireless services, tele-com, mobile computing and media applications have attracted great research interest for the development of new analog design automation software tools [1–9].

On the one hand, automation in circuit design has successfully demonstrated its usefulness, from circuit level design, for instance in [10–15], to system level designs, for example in [4,7,8, 16–19]. On the other hand, the current computers features (as speed and storage) have favored a transition from "hand-calculation" analog circuit design to a simulation-based electronic design automation (EDA) [20], approach. The key of this transition was the feasibility to include SPICE-like simulators within the loop of any circuit design optimization problem [19]. These EDA tools have been catalogued as "post-SPICE" tools [21], and their results are considered trustworthy due to the fact that they come directly from a SPICE like simulator.

The use of EDA tools for analog circuit design, opens the possibility of "human-computer collaboration" [19] that brings benefits as feedback for knowledge extraction allowing the designers to make better decisions [22], and to understand the interaction among design parameters, performances and constraints that a device has to accomplish.

1.1 Analog Circuit Optimization Tools: Categories and Classifications

Analog EDA tools are devoted to work out with several tasks such as circuit synthesis, design knowledge, multitopology selection or circuit optimization, among others. Analog circuit optimization consists in selecting a topology then finding the variables design values (V_{DD} , I_{BIAS} , W , L , etc.) to accomplish the circuit target requirements.

These sort of EDA tools, according to their implementation can be classified in six categories [23] :

1. **DESIGN KNOWLEDGE:** usually is a single-objective optimization performed by deterministic methods; then one solution is found. Its task is accomplished relatively fast and it is a quite useful tool to achieve insight as shown in [24]. It is necessary to highlight the capability of this method to find the complicated relations among design performances. Finally, the extracted expert knowledge is available to any designer for multiple purposes [22].
2. **LOCAL UNCONSTRAINED OPTIMIZER:** in this case, a sizing problem is reduced to a mathematical unconstrained cost (scalar) function that needs to be minimized. The key is to apply terms including penalties depending on the design parameters and/or performances. The problem can be solved by deterministic methods obtaining a single solution as in [25]. It is possible to point out two disadvantages: its success depends on an initial point and is a fast method for small sizing problems.
3. **CONSTRAINED OPTIMIZATION:** the sizing problem is translated into a constrained optimization problem as in [26], in this manner there exists the possibility of applying techniques as Genetic Programming capable to find a global optimum. This method has drawbacks such as the need to build models, nevertheless for big problems are not practical and decrease execution speed.

1.1. ANALOG CIRCUIT OPTIMIZATION TOOLS: CATEGORIES AND CLASSIFICATIONS3

4. GREEDY STOCHASTIC OPTIMIZATION: for this method a random search is used to find an optimum solution, then it needs to be guided for other methods as design knowledge or by using a behavior memory along the process. Due to its random nature, to find a global optimum is not guaranteed, but it has the capability to find a solutions set. An example of this method extended to multi-objective optimization is found in [27].
5. ANNEALING: this is a powerful tool to solve optimization problems by using statistical techniques to select the best solution into a solutions neighborhood, it can handle multi-objective problems and constraints. Some approaches include the variability into the design process and handle discrete values for the design variables as in [28]. Also, they have the capability to implement an up-hill method to scape from a local minima providing memory to the system . Mixed with other techniques, it has shown its usefulness in [12].
6. EVOLUTION: as in [29], this is a formal stochastic method which allows to handle multi-objective problems including design constraints by using a cost function . The best solutions are selected by the Pareto dominance, at the same time saving a history of the optimization process to avoid lost of the global optimum. Genetic operators are the key to explore a wide search space preventing that solutions be trapped in a local optimum.

In addition, independently from the implementation, it is possible to make another classification based on the performances estimation, which determines the way how a circuit will be treated and from this decision depends the relation speed vs. accuracy. Generally, there exist two performance estimation possibilities with their own variants as depicted in Figure 1.1 [19,30] :

1. STATIC: in this case, the circuit or system will be replaced by a model (mathematical equation or a regression model), which can be handled without the need to include a circuit simulator, as a result it is possible to save time. To build the model it is possible to make it by hand or automatically by using: symbolic analysis, neural networks, genetic programming, among others. Finally, this implementation is always compared with a circuit simulator to verify the solution really is correct.

2. DYNAMIC: for this approach it is necessary to use a circuit simulator but it is possible to chose between SPICE-like simulators (standards or high-capacity circuit simulators) and behavioral simulators (VERILOG-A for instance).

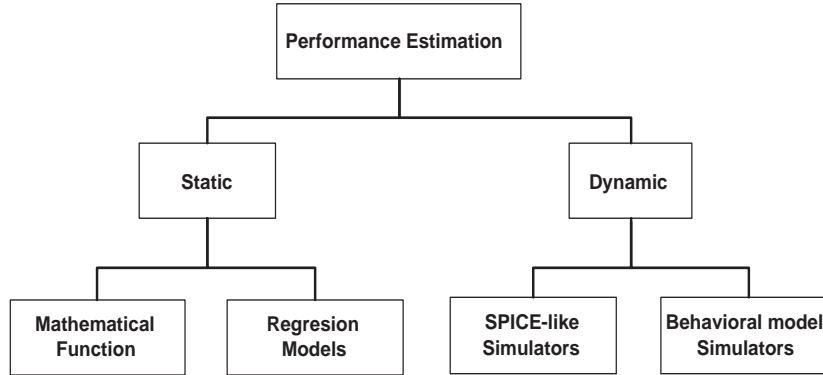


Figure 1.1: Performances Estimations [19].

This work proposes an analog circuit optimization methodology by using evolutionary algorithms and HSPICE[™] to extract the circuits performances. In this manner, it is possible to catalogue it as an evolutive methodology with a dynamic performance estimation.

1.2 EDA tools

Since 1980's EDA tools [25, 31–37] have made easier the IC design task, besides all of them were based on analytical equations. It is the case of IDAC [31] that performs a worst-case based design to size a library of analog schematics as: transconductance, operational and low-noise amplifiers, voltage and current references, oscillators and comparators. A collection of knowledge into an expert system which uses heuristic rules for making design decisions can be found in OP-1 [32] for the design of OPAMPs. OASYS [33] uses a hierarchical approach, breaking each design down into subblocks for handling a simpler set of equations and finally introducing a design space exploration. COARSE [34] achieves optimization of OPAMPs by using an iterative optimization approach by varying the DC operating point. Also, it is possible to find static performance estimation, as symbolic modeling (ISAAC) [36] to replace the circuit by analytic formulas for transfer functions avoiding to use a circuit simulator which is time-

consuming. However, the shortcoming was loss of accuracy.

Early's 1990s, arose the "first generation" analog design optimization tools. Among the first precursors is a static tool named OPTIMAN [38] that includes ISAAC for analytic modeling of the circuits and the optimization is based on annealing. This tool was tested on a folded-cascode OTA and on a switched-capacitor biquadratic filter, the optimized performances were: unity gain frequency, power, gain, phase margin, slew rate, noise, output range, input range and offset. SEAS [39] uses simulated evolution supported on design knowledge to optimize a two-stage OPAMP and optimizing: unity gain frequency, gain, slew rate, CMRR, phase margin, power and output range. Until then, the knowledge was the common resource as circuit libraries or design rules, while symbolic analysis was the other promising resource, but loss of accuracy is needed to simplify the analytical models, otherwise would be impractical to optimize the long terms.

During that decade, other tools that were proposed for local unconstrained optimizations by a corner analysis through a worst-case test as in SQP/EXCALIBUR [40], where a buffer amplifier is optimized in unity gain frequency, gain, phase margin, slew rate and voltage swing. However, that approach has two drawbacks, the first is an increase of the consumption time and the second one is that for the recent technologies there are more corners which needs to be simulated. Then, based on the worst-case idea, in [41] is used a concept called sensitivity band by defining the bounds between performances and parameters of the circuit; the variation of the parameters is made in terms of monotonicity, but due to its complexity, it is tested on linear circuits.

An annealing constrained system was proposed with a static performance estimation (ASTRX/OBLX) [42]. This time achieving to optimize a circuit in less time than before, due to instead using SPICE-like simulator, it uses an asymptotic waveform evaluation. This system was tested on a few OTA variants, among them: simple, cascode, folded cascode, and two stage; always optimizing only up to two objectives and the rest specifications are included as constraints. Effectively, this system optimizes in a reasonable time, preserving a reasonable prediction error too; unfortunately, the time and complexity setup, increases as the design variables increase.

Among the annealing dynamic approaches, there exists a multi-objective optimization of analog circuits in [28], and it handles discrete values for the design values and proposes different levels of constraints and a variation-aware technique based on the sensitivity of the objective functions to include them into the optimization loop, showing the usefulness of these approaches by optimizing two OPAMPs, a comparator and an analog buffer.

Another stochastic dynamic methodology named ANACONDA [27] was proposed by using stochastic search mixed with evolutionary algorithms concepts; also a SPICE-like is included simulator into the optimization loop and the system was tested on three OPAMPs. In this case there are only two optimization objectives: area and power; but were taking as constraints: gain, phase margin, noise, CMRR, PSRR, settling time and total harmonic distortion.

In [43] the simulated-based optimization is replaced with the use of transfer functions previously synthesized with a regression technique. An operational amplifier and a state variable filter were tested with this method but the main disadvantages are that for each circuit it is necessary to apply a properly regression technique and the global optimization algorithm can not handle discontinuous transfer functions.

Continuing with regression techniques and support variation methodologies, in HOLMES [44] are captured the relationships between design variables and performances. In this manner, it includes statistical process variations by creating regression polynomial models and making a statistical variation optimization from a distributed probability density functions; an OTA Miller was tested with this method and the optimization performance was power-area with unity gain, slew rate, gain and input-referred noise density as constraints.

Formal evolutionary systems began to appear in scene with the new century, for instance WATSON [45], where are generated new sets of design variables using SPICE as simulator to evaluate the performance of the given circuit and using a genetic multiobjective algorithm. The performances (operating point, small-signal, noise and/or transient analysis) are extracted from the simulations and passed to a next optimization stage using accurate reduced-order models to make a fit to a Pareto front. Two OTA are compared and optimized; the design variables were widths and lengths of the transistors, the biasing current and compensation capacitance; regarding to objectives, were optimized in: input-referred noise density, unity gain frequency,

gain and slew rate, taking into account saturation condition in transistors and limits on the gate-source overdrive as constraints.

Since 2005, the optimization tools have achieved reasonable times for dynamic systems, and accuracy for static ones. Since then, this kind of EDA tools have addressed new challenges: expert knowledge extraction, variation-support, multi-topology and system level optimization.

Regarding expert knowledge extraction it is possible to find dynamic tools as CAFFEINE [24, 30]. In this case, SPICE simulations are fitting into symbolic models (depending of current bias, drain-source voltage and source-gate voltage) by using genetic programming. An OTA was tested finding symbolic expressions for gain, phase margin, unity gain frequency, offset and slew rate. Then, it is possible to gain insight in a circuit without investing a lot of months of work and understand the trade-offs among the transistors and the rest of elements that form a circuit.

Multi-topology synthesis was applied before in [46, 47], but in MOJITO [48], there were incorporated 3,528 topologies which are optimized to select the best according to the desired specifications by using a multi-objective algorithm based on elitism. This tool is improved by using another evolutionary algorithm (EA) based in decomposition and adding more topologies being a total of 101,904 and was tested on an OTA by optimizing gain, unity gain frequency, dynamic range, slew rate, power, phase margin and area.

Until that moment, the optimization was performed in a “flat” fashion, that is to say, all transistors at once. Then, emerged the system level optimization which is devoted to optimize large circuits. There exists a bottom-up approach that consists of identifying fundamental blocks into a large circuit; then by a hierarchical decomposition it is possible to optimize each one of the components by using an evolutionary algorithm and afterwards, combining all of them as in [7, 17].

Other alternative to optimize large circuits, is a top-down approach as in [8], that begins with a language-based description giving the desired circuit functionality. The next step is to represent that description in a graph that allows to create one or more architectures which has the same functionality behavior and optimizing them. As in bottom-up and in top-down approaches, EAs are used as optimization engines.

About variation-support optimization a natural way to include the variations in the optimization process is by including a Monte Carlo analysis to each optimized solution as in [49], unfortunately this is impractical because the consumption time increases considerably. A trustworthy analog EDA tool needs to offer robustness solutions enforcing the optimization tasks with reasonable execution times, then the efforts have been focused to employ methods to avoid expensive times without loss of accuracy [26, 50, 51].

Continuing with this trend, Kriging models [5], are used to address a statistical performance for deterministic functions into stochastic process; two circuits were tested: LC oscillator and a two-stage operational amplifier. Other example is Multi-yield Pareto fronts [6] that works with the same methodology but applied to optimize a PLL.

Nevertheless, Monte Carlo based methodologies have been summarized, for instance in [20, 52] that use Pareto surfaces, but this time, reducing the number of evaluations required in Monte Carlo analysis ensuring accuracy by applying a Latin Hypercubes sampling.

1.3 Justification

Nowadays, hand-crafting analog design has to respond to the time-to-market constraints by encouraging the industry to grow and improve analog EDA tools [3, 9]. Such tools, increase productivity not only by reducing design time but also by diminishing the error-prone to which a manual process is subject. Industrial circuit design requires not only optimized designs, but also requires trustworthy and robustness to variations process. In this manner, the designer needs to handle a large number of variables, often against each other, to consider all these issues and performance requirements [4]. In this way, automated design of analog circuits has benefits for industrial design process improving productivity mainly by reducing the design time as shown in [3, 16, 30].

The academic efforts in this field have been fostered to the industry to use these methodologies. This is the case of *Virtuoso NeoCircuit* [53], *Circuit Explorer* [54], *MunEDA* [55], *Titan* [56] and *Arsyn* [57], among others. In general, all of them include optimization tools, which offers to accelerate the design process allowing designers to focus on creative tasks instead of spending time in repetitive tasks. These tools are capable to optimize a circuit with

deterministic or stochastic methods taking into account constraints, some of them including multi-objective optimization and including corner analysis.

Despite the different challenges in EDA tools, it is possible to find recent efforts to enhance the optimization task, especially by using EAs. Typically, EAs work with a set of non-dominated solutions which are a powerful way to analyze data; because, unlike to single solution methods, it is possible to identify and explore the trade-offs among the optimized objectives. A great number of optimization EDA tools use an EA due to its high capability to handle many variables and objectives taking into account constraints. Also, an EA is able to find optimal Pareto fronts, at the same time, saving all the optimization process to reuse for post analysis without having to resimulate. Some EAs, do not need much setup effort to scale the number of variables, objectives and constraints. All these issues, are attractive features that have been made that EAs achieve opportunities in analog circuit design optimization. From the state of art, it is possible to see how EAs recently have leaded the analog design optimization tools, then using them into optimization process, offers a promising field.

There are factors that have shown a marked viability on the development of automated analog circuit design such as success of previous work on this field, constant improvement of heuristics techniques and more computers capabilities such as speed and storage. In [4, 21, 44, 58] are agreements in the challenges and opportunities that each design problem presents. Also, it is claimed that we are at the beginning of the evolution of the post-SPICE analog tools, and is expected more improvements and accurate in circuit and level system.

The fraction of circuits that meets the specifications in a system among all the fabricated circuits is called “*yield*” [20]. While a $0.35\mu m$ technology has a yield around 95%, a $90nm$ technology has around 50% of yield for OTA's, filters, integrators and comparators that exhibit the similar performance levels [9, 58]. The relation between the performances and variation of parameters of a circuit generally is non-linear but it is possible to simulate it [51]. Under these conditions, a designer needs new tools to handle all the involved parameters in a circuit [9, 59] and for manufacture process, the optimization helps the designer to make high-performing designs [6].

In this manner it is possible to identify the benefits of using optimization analog circuit

design tools, such as:

- To validate if a design works properly, fulfilling the requirements, performances and trade-offs.
- To verify if a system architecture or selected topology is correct and meets specifications.
- To experiment the circuit to know what are the design limits before failing, taking into account more than one performance and/or constraint at the same time.
- To explore what elements are more sensitive and if there are issues which have risk to design.
- To make better decisions based on performance/contraints and the developed knowledge by using these tools.
- To enhance the circuit robustness by guaranteeing that the found solutions support process variation.
- To enhance circuits which have been previously hand crafted and have not reached their best performances.

1.4 Objectives

The main goal is to propose an EDA methodology for analog circuit optimization by applying EAs, encoding automatically the circuits and including support variation.

The proposed optimization will be based on HSPICE[™] simulations and compiled on a open source language code with the aim to be portable over the different operating systems.

It is possible to summarize the following objectives:

- Getting highest-quality circuit performance tradeoffs.
- Minimizing computational effort in the analog circuit optimization.
- Maximizing robustness by including circuit variation-aware.
- Exploring the behavior of the EAs with varying conditions of the genetic operators.
- Calibrating and comparing some evolutionary algorithms.
- Testing of the EAs with similar test functions including constraints.
- Apply graph theory for the automatic biasing of the circuit under optimization.

- Analyze a variation-support strategy to enhance the solution feasibility.

1.5 Thesis Organization

This thesis is organized as follows. The second chapter is devoted to outline the basic concepts about evolutionary algorithms, describes the multi-objective problem, the Pareto dominance and the genetic operators used along this work. At the end of that chapter, there are exposed three evolutionary algorithms to solving multi-objective problems by including constraints: Non-Dominated Sorting Genetic Algorithm (NSGA-II), Multi-Objective Evolutionary Algorithm by Decomposition (MOEAD) and Multi-Objective Particle Swarm Optimization (MOPSO) and a brief comparison among them are made by optimizing mathematical functions with Simulated Binary Cross-Over (SBX) and Differential Evolution (DE) as recombination operators.

The third chapter is devoted to show the circuit optimization methodology proposed in this Thesis based on evolutionary algorithms. Next, is performed a comparison between SBX and DE when are used in the optimization of eleven objectives of two mixed mode analog circuits: a Positive-type Second Generation Current Conveyor ($CCII^+$) encoded only with two variables and a Negative-type Second Generation Current Conveyor ($CCII^-$) encoded with eight variables. The chapter ends showing the optimization of nine amplifiers all of them with different number of design variables and eleven objective functions.

A new current-branches-bias assignment approach is proposed in the fourth chapter with the aim to accelerate the sizing process of analog integrated circuits composed of MOSFETs. This methodology is used to initialize the evolutionary algorithms in the optimization of two amplifiers: a Folded Cascode (FC) Operational Transconductance Amplifier (OTA) encoded with seven variables and a Recycled Folded Cascode (RFC) OTA encoded with ten variables. The examples show a reduction of generations to find optimal solutions and increased the number of biased solutions in less time comparing with the same optimization without using the proposed methodology.

The fifth chapter outlines the yield and tolerance concepts for analog circuits. Next, there are summarized the variation analysis approaches and there are grouping them in: worst case and non-worst case approaches. Next, it is shown a Worst Case approach based on sensitivity and a Non-WorstCase based on Monte Carlo simulation to broad the variation aware optimization of analog circuits. A complete multi-objective optimization system is presented to perform these

approaches, it is able to finding optimal solutions taking into account the fabrication process variations.

The sixth chapter summarizes the conclusions around this work. Appendix A is devoted to show the circuit optimizer software tool developed to perform all the circuit optimizations in this work. In Appendix B are listed the transistor models to optimize all the circuits in Chapters 3 to 5.

Chapter 2

Evolutionary Algorithms

2.1 Introduction

This chapter consists of an outline about Evolutionary Algorithms (EA), first by defining the main terms that are used continuously in this field, and second by describing the general procedure of an EA. Next, are described three EAs and a brief comparison among them are made by optimizing ZDT functions with SBX and DE as recombination operators.

2.2 Evolutionary Algorithms concepts

2.2.1 Individuals, Population, Evolutionary Operators and Objective Function

An *individual* represents an encoded solution to some specific problem. Each individual is defined by a biological *genotype*, when the genotype is decoded (for instance to represent a specific circuit) then is named *phenotype*. A genotype is conformed by one or more *chromosomes*. Such chromosomes in turn are composed of *genes* that have certain values named as *alleles*. A *locus* is the position that an allele has within the chromosome. Figure 2.1 [60] depicts a chromosome, that consists of n genes and each one has a specific value (allele) located in a specific position (locus).

When an individual is represented by only one chromosome, the words: individual and chromosome, are used to referring as the same; a set of individuals (or chromosomes) yield a

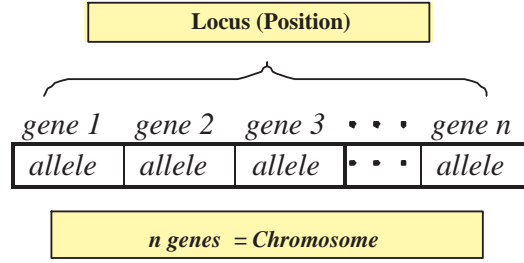


Figure 2.1: A chromosome example [60]

population. Figure 2.2 shows a population which consists of N individuals, and each individual consists of n genes.

N Individuals (Population)

	1	2	3	• • •	n
1	0.2	0.4	0.7	• • •	0.5
2	0.8	0.9	0.3	• • •	0.6
3	0.1	0.5	0.9	• • •	0.4
•	•	•	•	•	•
•	•	•	•	•	•
•	•	•	•	•	•
N	0.6	0.2	0.1	• • •	0.8

Individual 1, gene 1 = 0.2

↑

Figure 2.2: Population example.

Equation (2.1) defines a population \mathbf{P} as the individuals set $\mathbf{P} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ where \mathbf{x}_i represents the i -th individual composed by n genes ($\mathbf{x}_i = \{x_1^i, x_2^i, \dots, x_n^i\} | 0 \leq i \leq N$).

$$\mathbf{P} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_n^1 \\ x_1^2 & x_2^2 & \dots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^N & x_2^N & \dots & x_n^N \end{bmatrix} \quad (2.1)$$

Then the EA procedure tries emulating the nature behavior, from parents generating new offspring with best fitness than the previous one. The *evolutionary operators* are the responsible

of this process, and such operators are three:

1. *Mutation*: consisting of selecting a parent to change the value of an allele choosing randomly a locus.
2. *Recombination*: for instance, cross-over is a form of recombination and consists of selecting parents (usually two) and each one is cut and recombined with the other part of the other parent.
3. *Selection*: This process is the responsible to choose among all the parents and the offspring, such those that have the best fitness.

Then \mathbf{P} is the set of individuals which conforms a population in the generation t , and $|\mathbf{P}|$ denotes the population size. It is possible to render the next population (\mathbf{P}^{t+1}) from the current population (\mathbf{P}^t), by using evolutionary operators: μ_r denotes recombination, μ_m denotes mutation and μ_s denotes selection evolutionary operators. The individuals in the current population (\mathbf{P}^t) are called “parents” and the individuals in the next population (\mathbf{P}^{t+1}) are called the offspring.

An *objective* function is a feature of the problem domain and defines the EA’s optimality condition. The *fitness* function allows to measure with a real-value a solution based in how much satisfies the objective function(s).

2.2.2 The General Evolutionary Algorithm

The evolutionary task begins when an initialization procedure generates (usually randomly) a population of individuals yielding the first parent population (\mathbf{P}^t , the first generation is denoted by $t = 0$); next, by using evolutionary operators (μ_r , μ_m and μ_s) a new population (\mathbf{P}^{t+1}) is generated yielding the offsprings (usually $|\mathbf{P}^t| = |\mathbf{P}^{t+1}|$). Afterwards, a fitness function evaluation (from the objective functions) is performed for each new individual (\mathbf{x}^{t+1}) in the new population (\mathbf{P}^{t+1}), if the offsprings have better fitness than their parents, then the parents are replaced by their offspring. Finally, a stop criterium (ξ) decides when the task should stop from a set of parameters μ_ξ . All this process is summarized in the Algorithm 1 [61].

Algorithm 1 General Pseudocode for an Evolutionary Algorithm

```

1:  $t \leftarrow 0$ 
2:  $\mathbf{P}^t \leftarrow \text{initialize}$ 
3:  $\mathbf{F}^t \leftarrow \text{evaluate } \mathbf{P}^t$ 
4: repeat
5:    $\mathbf{P}^{new} \leftarrow \text{recombine } (\mathbf{P}^t, \mu_r)$ 
6:    $\mathbf{P}^{new} \leftarrow \text{mutate } (\mathbf{P}^{new}, \mu_m)$ 
7:    $\mathbf{F}^{t+1} \leftarrow \text{evaluate } (\mathbf{P}^{new})$ 
8:    $\mathbf{P}^{t+1} \leftarrow \text{select } (\mathbf{P}^t, \mathbf{P}^{new}, \mathbf{F}^t, \mathbf{F}^{t+1}, \mu_s)$ 
9:    $t \leftarrow t + 1$ 
10: until  $(\xi(\mathbf{P}(t), \mu_\xi) = \text{true})$ 

```

2.3 Multiobjective Optimization and Pareto Dominance

2.3.1 Multiobjective Design Problem

Lets us consider a multiobjective design problem of the form [62]¹:

$$\begin{aligned}
 & \text{minimize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_M(\mathbf{x}))^T \\
 & \text{subject to } h_l(\mathbf{x}) \geq 0, \quad l = 1 \dots p, \\
 & \text{where } \mathbf{x} \in X.
 \end{aligned} \tag{2.2}$$

where $X \subset \mathbb{R}^n$ is the decision space for the design variables, $\mathbf{x} = (x_1, \dots, x_n)$ is called the decision vector. $\mathbf{f}(\mathbf{x})$ is the performance objective vector, $f_j(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$, $m = 1 \dots M$ ($M \geq 2$) are performance objective functions and $h_l(\mathbf{x})$, $l = 1 \dots p$, are performance constraints.

While, objectives are used as a necessary improvement condition to regard a solution best than other ones, constraints are used as quality measures that allows to identify which solutions accomplishes in a best way the trade-offs.

Regarding to circuit sizing, each variable x represents a design variable as the width (W) or length (L) of the MOSFETs. The circuit design task consists of finding the nominal design variables values (\mathbf{x}) which accomplish the specified performances ($\mathbf{f}(\mathbf{x})$) and carried through constraints(h).

¹It is possible to consider to maximize instead to minimize the function

2.3.2 Pareto Dominance

It is possible to define the Pareto dominance [63, 64] as $\mathbf{x}_a < \mathbf{x}_b$ (\mathbf{x}_a dominates \mathbf{x}_b) if all $f_m(\mathbf{x}_a)$ in $\mathbf{f}(\mathbf{x}_a)$ are equal or better than all $f_m(\mathbf{x}_b)$ in $\mathbf{f}(\mathbf{x}_b)$ and at least one $f_m(\mathbf{x}_a)$ is better than $f_m(\mathbf{x}_b)$ (for $m = 1, \dots, M$), where better means less when the objective is to minimize and high when the objective is to maximize.

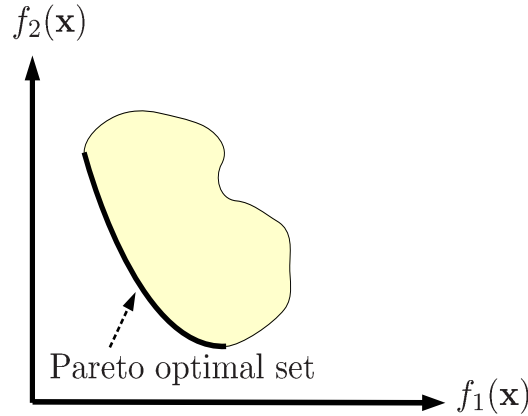


Figure 2.3: Pareto optimal set example for two objective functions.

Very often, since the objectives in (5.11) contradict each other, no point in X minimizes all the objectives simultaneously. One has to balance them, and the best tradeoffs among the objectives can be defined in terms of Pareto optimality. In this manner, a solution is considered as optimal if it can not be improved without deterioration to at least one of its components; then it is probable that there will be more than one Pareto optimal solution and the multiobjective optimization problem finishes when the Pareto optimal set is found. Figure 2.3 shows the Pareto optimal set of a given solution set considering a minimization problem with $M = 2$, $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}))$.

The Pareto dominance can take into account the constraints of a problem. A solution a is said to be constrained-dominated in a solution b , if any of the following conditions is true [65]: solution a is feasible and solution b is no, solutions a and b are both infeasible, but solution a has a smaller overall constraint violation or, solutions a and b are feasible and solution a dominates solution b .

To comparing two different approximations to a given Pareto front, usually it is used a “cov-

erage metric” [66]. Let A and B be two approximations to the Pareto front of a multi-objective problem. $C(A, B)$ is defined as the percentage of the solutions in B that are dominated by at least one solution in A [65]:

$$C(A, B) = \frac{|\{u \in B \mid \exists v \in A : v \text{ dominates } u\}|}{|B|} \quad (2.3)$$

$C(A, B)$ is not necessarily equal to $1 - C(B, A)$. $C(A, B) = 1$ means that all solutions in B are dominated by some solutions in A .

2.3.3 Diversity and Efficiency

When we have a problem to solve, there may be several suitable algorithms available. We would obviously like to choose the best, in such a manner, it raises the question of how to decide which is preferable. Besides the solution convergence when the optimization experiment is repeated, there exists two features that allow to compare among algorithms.

An important EA feature is *diversity* [60], which consists of avoiding that the entire population converges to a single point ignoring the rest of the search space. It is desirable to preserve diversity and the convergence of the solutions to the Pareto front at the same time, when a experiment is repeated trough different runs.

Also, it is possible to define the *efficiency* of an algorithm as simply how fast it runs, then it is necessary to express the unit for the theoretical efficiency of an algorithm, as the time taken by an algorithm within a multiplicative constant. This concept works regardless the programming language, the compiler used, the skill of the programmer and the implementation hardware [67].

We usually do not know the problem size beforehand, and either, if all problems require the entire range of functions in the algorithm. Then, it is considered an asymptotic behavior of the algorithm for a very large problem size, this behavior is expressed in an *Asymptotic Notation* [68]. Among the most important asymptotic notations is the “Big Oh”² notation that is a mathematical symbol to denote: “*the order of*”.

²There exists other notations as omega, theta and little oh.

2.4 Genetic Operators

The genetic operators are used in EAs in order to recombine existing individuals (or solutions) of the current generation to render a new one individual. A genetic operator helps along the optimization procedure, to converge to the Pareto front and to preserve diversity, then the success of the optimization largely depends on these operators.

The basic genetic operators are *crossover* and *mutation* [69] but exist other operators as Simulated Binary Cross-Over Operator (SBX) [70] and Differential Evolution (DE) [71] which have shown to improve the performance of basic operators.

2.4.1 Crossover and Mutation

The crossover operator yields a new individual by swapping the genes at random between the chromosomes of two parents. Usually, This process is called *single-point crossover* when is chosen a gene of a parent chromosome as swap point, and all the genes after or before that swap point are replaced for the genes of the other parent chromosome. Figure 2.4 depicts an one-point crossover example for $n = 5$. There exists other variants as *two-points cross over*, *cut-splice crossover*, *uniform crossover*, among others.

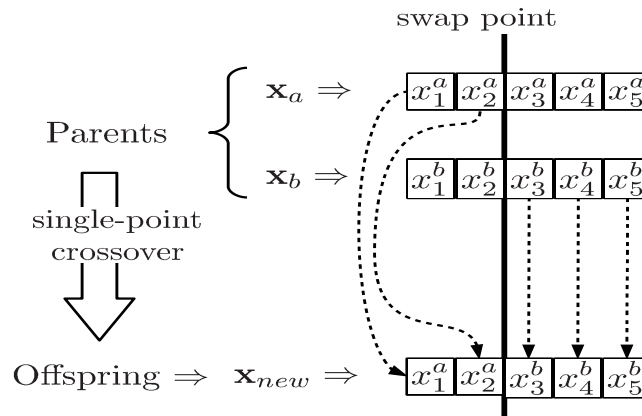


Figure 2.4: Single-point crossover example for $n = 5$.

While crossover operator yields new individuals by recombining the chromosomes and preserves the genes, the mutation operator modifies slightly one gene of the chromosomes in a randomly fashion with the aim to preserve the diversity. Both, the probability for applying

mutation and the variation over the gene, should be low. Figure 2.5 shows an example of the mutation operator that has randomly chosen the third gene.

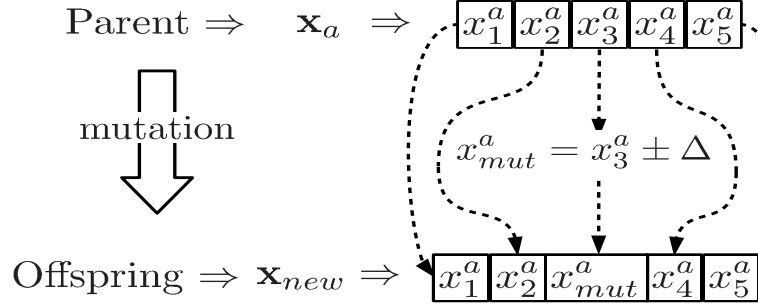


Figure 2.5: Mutation example for the third gene.

2.4.2 Differential Evolution (DE)

The Differential Evolution (DE) operator to improve convergence and diversity [14, 72], consists of randomly choosing three parents: \mathbf{x}_a , \mathbf{x}_b and \mathbf{x}_c from a population \mathbf{P} . A new solution $\mathbf{x}_{new} = x_1^{new}, x_2^{new}, \dots, x_n^{new}$ is generated as shows (2.4) [71]:

$$x_k^{new} = \begin{cases} x_k^a + R \cdot (x_k^b - x_k^c) & \text{if rand() } < C, \\ x_k^a & \text{otherwise,} \end{cases} \quad \text{for } k = 1, 2, \dots, n. \quad (2.4)$$

Where R is a constant factor which controls the amplification of the differential variation, C is the cross-over probability and rand() is a function that returns a random real number in the interval $[0, 1)$. Recommended values for these constants are $R \in [0.5, 1.0]$, $C \in [0.8, 1.0]$ [73].

2.4.3 Simulated Binary Cross-Over Operator (SBX)

This operator has shown that overcomes some issues in the recombination process [65, 70, 74] and it uses a probability density (Eq. 2.5) as function of the *spread factor* (β_k). β_k is defined as the ratio of the spread of the offspring to that of the parent points for the k -th gene.

$$P(\beta_k) = \begin{cases} 0.5(\eta + 1)\beta_k^\eta & \text{if } \beta_k^\eta \leq 1 \\ 0.5(\eta + 1)\frac{1}{\beta_k^{\eta+2}} & \text{otherwise} \end{cases} \quad (2.5)$$

The η is an integer value called *distribution index*. This operator allows creating two offsprings at the same time as shows (2.6) [65], where \mathbf{x}_a and \mathbf{x}_b are the parents and \mathbf{x}_{new1} and \mathbf{x}_{new2} are the offspring.

$$\begin{aligned} x_k^{new1} &= 0.5[(1 + \beta_q) \cdot x_k^a + (1 - \beta_q) \cdot x_k^b] \\ x_k^{new2} &= 0.5[(1 - \beta_q) \cdot x_k^a + (1 + \beta_q) \cdot x_k^b] \end{aligned} \quad (2.6)$$

β_q is defined as (2.7) where ρ is a random real number in the interval $[0, 1)$.

$$\begin{aligned} \rho &= rand() \\ \beta_q &= \begin{cases} (2\rho)^{\frac{1}{\eta_c+1}} & \text{if } \rho \leq 0.5 \\ \left(\frac{1}{2(1-\rho)}\right)^{\frac{1}{\eta_c+1}} & \text{otherwise} \end{cases} \end{aligned} \quad (2.7)$$

2.4.4 Polynomial Mutation

Polynomial mutation [65, 75] as SBX, uses a specific probability density shown in (2.8) where ρ is a random real number in the interval $[0, 1)$ and η_m is an integer number.

$$\delta_k = \begin{cases} (2\rho)^{\frac{1}{\eta_m+1}} - 1 & \text{if } \rho < 0.5 \\ 1 - [(2\rho)^{\frac{1}{\eta_m+1}}] & \text{otherwise} \end{cases} \quad (2.8)$$

The mutation can be performed as (2.9), where x_k^L and x_k^U are the lower and upper bounds values for the k -th gene, respectively, and \mathbf{x}^{new} is a given individual in the next generation.

$$x_k^{new} = x_k^{new} + (x_k^U - x_k^L) \cdot \delta_k \quad (2.9)$$

2.5 NSGA-II, MOEAD and MOPSO

2.5.1 Non-Dominated Sorting Genetic Algorithm II (NSGA-II)

This is an improved version of a previous NSGA algorithm by including elitism and was named as NSGA-II. Algorithm 2 [76, 77] summarizes the NSGA-II procedure and its efficiency is $O(mN^2)$, where m is the number of objectives and N is the population size. NSGA-II approximates the Pareto Front of a MOP by sorting and ranking all solutions in order to choose the better solutions to make a new offspring, this means, by ranking all the population in different Pareto subfronts that it will be possible to know which solutions show better performance.

Algorithm 2 NSGA-II Algorithm

```

1:  $P_0$ =random,  $Q_0$ =random
2:  $t=0$ 
3:  $P_{t+1} = \emptyset$  and  $i = 1$ 
4: repeat
5:    $R_t = P_t \cup Q_t$ 
6:    $F = \text{fast-non-dominated-sort}(R_t)$ 
7:   crowding-distance-assignment( $F_i$ )
8:   repeat
9:      $P_{t+1} = P_{t+1} \cup F_i$ 
10:     $i = i + 1$ 
11:  until  $|P_{t+1}| + |F_i| \leq N$ 
12:  Sort( $F_i, <_n$ )
13:   $P_{t+1} = P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$ 
14:   $Q_{t+1} = \text{make-new-pop}(P_{t+1})$ 
15: until stop criteria

```

In this algorithm is contemplated a way to choose the best solution between two solutions in the same subfront preserving diversity, in this form it is possible to select the best part of a population without losing diversity.

Then NSGA-II is based on two main procedures : *Fast Nondominated Sort* and *Crowding Distance Assignment*. These two procedures ensure elitism and it is possible to add constraints

to ensure that the solutions are feasible [76].

At the beginning, it is necessary to randomly initialize the parameters and start by generating two populations (P_o and Q_o) each one of size N , from random values into a feasible region. The NSGA-II procedure in each generation consists of rebuilding the current population (R_t) from the two original populations (P_t and Q_t) then the new size of current population will be $2N$.

Now through a nondominated sorting all solutions in R_t are ranked, and classified in a family of subfronts. In the next step is necessary to create from the current population R_t (previously ranked and ordered by subfront number) a new offspring (P_{t+1}), the objective will be to choose from a population of size $2N$, the N solutions which belong to the first subfronts. In this manner, the last subfront could be greater than necessary, then a measure ($i_{distance}$) is used to identify the better solutions and preserving elitism by selecting the solutions that are far the rest, this is possible simply by modifying a little bit the concept of Pareto dominance as follows:

$$i <_n j \text{ if } \left[(i_{rank} < j_{rank}) \text{ or } (i_{rank} = j_{rank}) \right] \text{ and } (i_{distance} > j_{distance})$$

Fast Non-Dominated Sort: Algorithm 3 shows this procedure which is responsible to rank each solution into a subfront, and starts by selecting the nondominated solutions among the current population (R_t). This first group of solutions will be labeled as the solutions into the first subfront (F_1) and are separated from R_t . For the remaining solutions in R_t are selected the nondominated solutions again but this time they are labeled into the second subfront (F_2) and separated from R_t like the solutions in (F_1) were separated before. This procedure continues until all solutions in R_t are ranked into a subfront.

The procedure uses a counter for each solution, such counter allows us to know how many solutions dominate to each solution (n_p where p is the p -solution). In the same way, there is a set which contains all the solutions dominated for each solution (all solutions in S_p are dominated by p -solution). First are taken the solutions with counter equal to zero and to each solution in their set of dominated solutions are diminished their counters in one. In this way the next subfront is composed by the remaining solutions with counter equal to zero. This continues until all solutions have been ranked. In Figure 2.6 there is an outcome example of

this procedure, where the real Pareto front is in solid red color, the first subfront is in circled blue color, and so on.

Algorithm 3 Fast Nondominated Sort

```

1: for each  $p \in P$  do
2:    $S_p = \emptyset$ 
3:    $n_p = 0$ 
4:   for each  $q \in P$  do
5:     if ( $p < q$ ) then
6:        $S_p = S_p \cup \{q\}$ 
7:     else if ( $q < p$ ) then
8:        $n_p = n_p + 1$ 
9:     end if
10:    if  $n_p = 0$  then
11:       $rank = 1$ 
12:       $F_1 = F_1 \cup \{p\}$ 
13:    end if
14:  end for
15: end for
16:  $i = 0$ 
17: while  $F_i \neq \emptyset$  do
18:    $Q = \emptyset$ 
19:   for each  $p \in F_i$  do
20:     for each  $q \in S_p$  do
21:        $n_q = n_q + 1$ 
22:       if  $n_q = 0$  then
23:          $rank = i + 1$ 
24:          $Q = Q \cup \{q\}$ 
25:       end if
26:     end for
27:   end for
28:    $i = i + 1$ 
29:    $F_i = Q$ 
30: end while
  
```

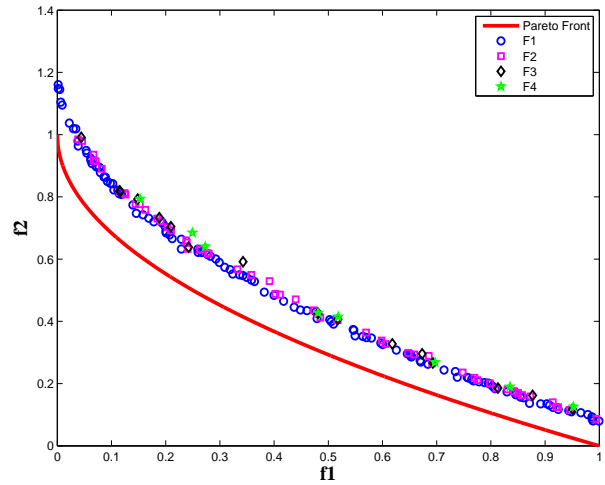


Figure 2.6: Fast Nondominated Sort Algorithm example

Crowding Distance Assignment: This is the second procedure to help to select solutions which will generate the offspring, and has sense where is necessary to choose the last members

Algorithm 4 Crowding Distance Assignment

```

1:  $l = |T|$ 
2: for each  $i$  do
3:   set  $T[i]_{distance} = 0$ 
4:   for each objective  $m$  do
5:      $T = \text{sort}(T, m)$ 
6:      $T[1]_{distance} = T[l]_{distance} = \infty$ 
7:     for  $i = 2$  to  $(l-1)$  do
8:        $T[i]_{distance} = T[i]_{distance} + (T[i+1] \cdot m - T[i-1] \cdot m) / (f_m^{max} - f_m^{min})$ 
9:     end for
10:  end for
11: end for

```

of the population P_{t+1} into a subfront, because all subfront members then have other ranking parameters into their subfront. The main idea is to perform a density estimation named *crowding distance* ($i_{distance}$) by sorting in ascending order the solutions for each objective function, then for each objective it is first selected the smallest and largest limit found and an infinite value is assigned to their crowding distances. Algorithm 4 shows the pseudocode for this procedure.

2.5.2 Multi-Objective Evolutionary Algorithm based on Decomposition (MOEAD)

The basic idea of MOEAD is the decomposition of a multiobjective problem in scalar optimization subproblems by a *weights vector*. This vector associates a weight (λ) for each subproblem which is considered as a single individual in the population which is going to try to improve by itself and to its nearby (*neighbors*).

After the initialization of the parameters the first step in MOEAD is related to define the N spread weights vector over the objectives space (to each individual corresponds one λ_i). One way can be by using a parameter H in a sequence as described by (2.10):

$$\left\{ \frac{0}{H}, \frac{1}{H}, \dots, \frac{H}{H} \right\} \quad (2.10)$$

In Algorithms 5 and 6 are depicted the pseudocode to generate these vectors, for three and m objectives respectively. It is necessary to chose a value for H and depending on this number the

population size is set.

Algorithm 5 Build spread of N weight vectors ($M = 3$)

```

1:  $i = 1$ , set  $H$ 
2: for  $\mu_1 = 0$  to  $1$ , step  $\frac{1}{H}$  do
3:   for  $\mu_2 = 0$  to  $1$ , step  $\frac{1}{H}$  do
4:     if  $1 - (\mu_1 + \mu_2) \geq 0$  then
5:        $\lambda_i = \{\mu_1 \ \mu_2 \dots (1 - (\mu_1 + \mu_2))\}$ 
6:        $i = i + 1$ 
7:     end if
8:   end for
9: end for

```

Algorithm 6 Build spread of N weight vectors for M objectives

```

1:  $i = 1$ , set  $H$ 
2: for  $\mu_1 = 0$  to  $1$ , step  $\frac{1}{H}$  do
3:   for  $\mu_2 = 0$  to  $1$ , step  $\frac{1}{H}$  do
4:      $\vdots$ 
5:   for  $\mu_{M-1} = 0$  to  $1$ , step  $\frac{1}{H}$  do
6:     if  $1 - (\mu_1 + \mu_2 + \dots \mu_{M-1}) \geq 0$  then
7:        $\lambda_i = \{\mu_1 \ \mu_2 \dots (1 - (\mu_1 + \mu_2 + \dots \mu_{M-1}))\}$ 
8:        $i = i + 1$ 
9:     end if
10:   end for
11:    $\vdots$ 
12: end for
13: end for

```

Therefore, it is possible to define a number (T) of neighborhoods for each λ_i and it is necessary to calculate the Euclidean distance between each λ_i ; finally for each λ_i is going to be (T) neighborhoods nearby and they will be saved in \mathbf{B}_i . Algorithm 7 shows the steps performed by MOEAD [66, 77] and its efficiency is $O(MNT)$ where M is the number of objectives, N the population size and T is the neighborhood size.

In each generation there is a population of N points $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N \in X$ where $\mathbf{x}_i = (x_1^i, x_2^i \dots x_n^i)$ is the current solution to the i -th subproblem and there are $\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_N$, where $\mathbf{f}_i = (f_1(\mathbf{x}_i), f_1(\mathbf{x}_i), \dots, f_M(\mathbf{x}_i))^T$ is the objectives vector and $f_m(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$.

In the procedure it is necessary to generate a new individual \mathbf{y} which will be compared with all its neighborhood by applying a decomposition approach ($g[\mathbf{x}_i \mid \lambda_i, \mathbf{Z}^*]$) such as the *Tchebycheff Approach* and each neighbor worse than this new individual will be replaced by it in an external population (EP) which is used to store non-dominated solutions.

In the Tchebycheff Approach, the scalar optimization problem is described by (2.11), where $\mathbf{Z}^* = \{z_1^*, z_2^*, \dots, z_M^*\}^T$ are the best current objective functions found [62].

$$g(\mathbf{x}_i \mid \lambda_i, \mathbf{Z}^*) = \max\{\lambda_i | f_m(\mathbf{x}_i) - z_m^* | \}_{1 \leq i \leq N, 1 \leq m \leq M} \quad (2.11)$$

Algorithm 7 MOEAD Algorithm

```

1: build an uniform spread of N weight vectors ( $\lambda$ )
2: for  $i = 1, 2, \dots, N$  do
3:    $\mathbf{B}_i = \{b_1^i, b_2^i, \dots, b_T^i\}$ 
4: end for
5:  $t = 1$ , POP=random(), set  $E = \emptyset$ ,  $T$ 
6: repeat
7:   for  $i = 1, 2, \dots, N$  do
8:     randomly select parents from  $\mathbf{B}_i$ 
9:     generate new individual  $\mathbf{y}$ 
10:    for each  $\ell \in \mathbf{B}_i$  do
11:      if  $g(\mathbf{y} \mid \lambda_\ell, \mathbf{Z}^*) \leq g(\mathbf{x}_\ell \mid \lambda_\ell, \mathbf{Z}^*)$  then
12:         $\mathbf{x}_\ell = \mathbf{y}$ 
13:         $\mathbf{f}_\ell = \mathbf{f}(\mathbf{y})$ 
14:      end if
15:    end for
16:  end for
17:  remove from EP all vectors dominated by  $\mathbf{f}(\mathbf{y})$ 
18: until stop criteria

```

2.5.3 Multi-Objective Particle Swarm Optimization (MOPSO)

In the Multi-Objective Particle Swarm Optimization Algorithm (MOPSO) there are N particles denoted by \mathbf{x}_i where $i = 1, 2, \dots, N$, and are represented by their positions in the search space. Each particle $\mathbf{x}_i = (x_1^i, x_2^i, \dots, x_n^i)$ represents a position in the space and depends of its previous local best value ($\mathbf{x}_{best,i}$), and the previous global best value (\mathbf{x}_{bestg}). To compute the speed of each particle the expression in (2.12) [78] is used.

$$V_k^i = k_w V_k^i + R_1(x_k^{(best,i)} - x_k^i) + R_2(x_k^{bestg} - x_k^i) \Big|_{\substack{1 \leq i \leq N \\ 1 \leq k \leq n}} \quad (2.12)$$

\mathbf{V}_i is the current velocity of the particle i -th, k_w is the inertia weight which takes typical values less than 1; $\mathbf{x}_{(best,i)}$ is the best position of particle i -th, \mathbf{x}_i is the position of the current particle and \mathbf{x}_{bestg} is a global best selected among the global best solutions. R_1 and R_2 are random

real numbers in the interval $(-1, 1)$. The new position of each particle is computed as Equation (2.13).

$$x_k^i = x_k^i + V_k^i \quad (2.13)$$

Algorithm 8 Pseudocode for MOPSO

```

1: Initialize NoLoops, bound limits, population ( $N$ ) and velocities
2: Evaluate population
3: for  $i=1$  to  $N$  do
4:    $\mathbf{x}_{(best,i)} \leftarrow \mathbf{x}_i$ 
5: end for
6: Select non-dominated particles and save them in REP
7: for  $t=1$  to NoLoops do
8:   for  $i=1$  to  $N$  do
9:      $\mathbf{x}_{bestg} \leftarrow$  Select randomly among solutions in REP
10:    Update particle velocity
11:    Update particle position
12:    Ensuring new position is into bound limits
13:   end for
14:   Evaluate population
15:   for  $i=1$  to  $N$  do
16:     Update  $\mathbf{x}_{(best,i)}$ 
17:   end for
18:   Select non-dominated particles and save them in REP
19: end for

```

Algorithm 8 shows the pseudo-code for MOPSO [79] and its efficiency is $O(mN)$, where m is the number of objectives and N is the population size.

In line 1 there is the initialization procedure where the bound limits are set, and the particles are initialized randomly inside these bound limits. The velocities are initialized with zero values. In line 2, the population is evaluated to update the best position ($\mathbf{x}_{best,i}$) at line 3, for each particle. Afterwards in line 6, the non-dominated particles are gathered into a repository (*REP*). The optimization procedure begins in line 7, and for each particle is selected a best

particle among the solutions in REP (line 9). Then the velocity and position of each particle is updated by using Equations (2.12) and (2.13), and avoiding going beyond the bound limits (lines 10 to 12). Once all the particles are updated, then an evaluation process is applied to update the best position for each particle and finally the non-dominated particles are selected and saved in a repository (REP)(lines 14 to 18). This process continues until a determined number of loops or until a stop criterium.

2.5.4 Behavior of NSGA-II, MOEAD and MOPSO on test functions ZDT

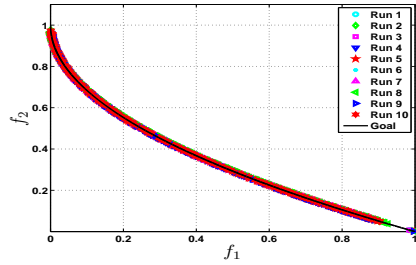
Zitzler *et al.* [80], provided a comparison of various evolutionary approaches to multiobjective optimization using six carefully chosen test functions, each one involves a particular feature, that is known to cause difficulty in the evolutionary optimization process, mainly in converging to the Pareto-optimal front. In Table 2.1, the four test functions used in this work are shown, where $ZDT1$, $ZDT2$, $ZDT3$ and $ZDT4$ are the name of the selected functions, n is the number of variables used for the functions, and bounds are the maximum and minimum search limits for each variable. For the four selected functions the goal is to minimize the functions given. The test function $ZDT1$ has a convex Pareto-optimal front, $ZDT2$ is the nonconvex contrapart to $ZDT1$. $ZDT3$ represents the discreteness feature, its Pareto-optimal front consist of several noncontiguous convex parts. $ZDT4$ contains 21^9 local Pareto-optimal fronts then it tests the EA's ability to deal with multimodality.

There were performed different experiments with the ZDT functions for NSGA-II, MOEAD and MOPSO by using two different evolutionary operators: DE and SBX. The aim is to show the usefulness of those three different evolutionary algorithms and their performance by using different evolutionary operators along ten runs to make an statistical study of the dominance of each one of the experiments compared with the others. All the runs were performed with a population size of 300 individuals for the ZDT functions listed on Table 2.1 over 1000 generations and along ten runs. The results for all these experiments are showed in Figs. 2.7-2.11 where the solid line depicts the goal for each ZDT function.

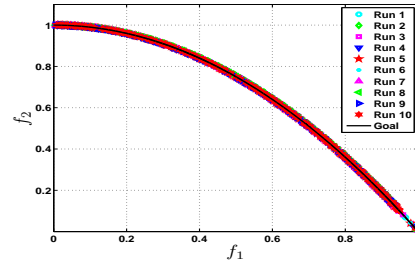
Finally in Table 2.2 are listed the statistical results for all these experiments with 5 variables, by calculating the coverage metric of each method (rows) on the other methods (columns). In

Table 2.1: Test Functions

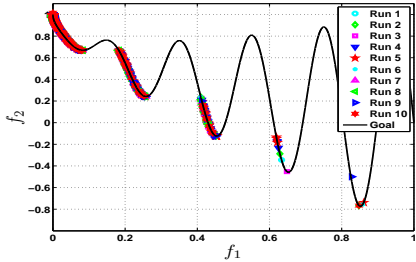
Function	n	Bounds	Functions	Optimal Sol.
$ZDT1$	15	$x_i \in [0, 1]$	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})[1 - \sqrt{\frac{x_1}{g(\mathbf{x})}}]$ $g(\mathbf{x}) = 1 + \frac{9}{n-1} \sum_{i=2}^n x_i$	$x_1 \in [0, 1]$ $x_i = 0,$ $i = 2, \dots, n$
$ZDT2$	15	$x_i \in [0, 1]$	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})[1 - (\frac{x_1}{g(\mathbf{x})})^2]$ $g(\mathbf{x}) = 1 + \frac{9}{n-1} \sum_{i=2}^n x_i$	$x_1 \in [0, 1]$ $x_i = 0,$ $i = 2, \dots, n$
$ZDT3$	15	$x_i \in [0, 1]$	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})[1 - \sqrt{\frac{x_1}{g(\mathbf{x})}} - \frac{x_1}{g(\mathbf{x})} \sin(10\pi x_1)]$ $g(\mathbf{x}) = 1 + \frac{9}{n-1} \sum_{i=2}^n x_i$	$x_1 \in [0, 1]$ $x_i = 0,$ $i = 2, \dots, n$
$ZDT4$	15	$x_1 \in [0, 1]$ $x_{2,\dots,n} \in [-5, 5]$	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})[1 - \sqrt{\frac{x_1}{g(\mathbf{x})}}]$ $g(\mathbf{x}) = 1 + 10(n-1) + \sum_{i=2}^n [x_i^2 - 10 \cos(4\pi x_i)]$	$x_1 \in [0, 1]$ $x_i = 0,$ $i = 2, \dots, n$



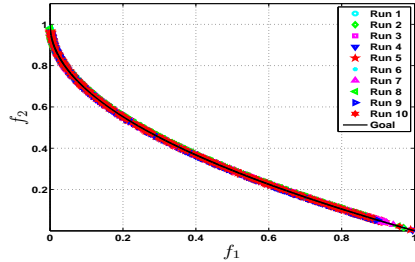
(a) ZDT1



(b) ZDT2

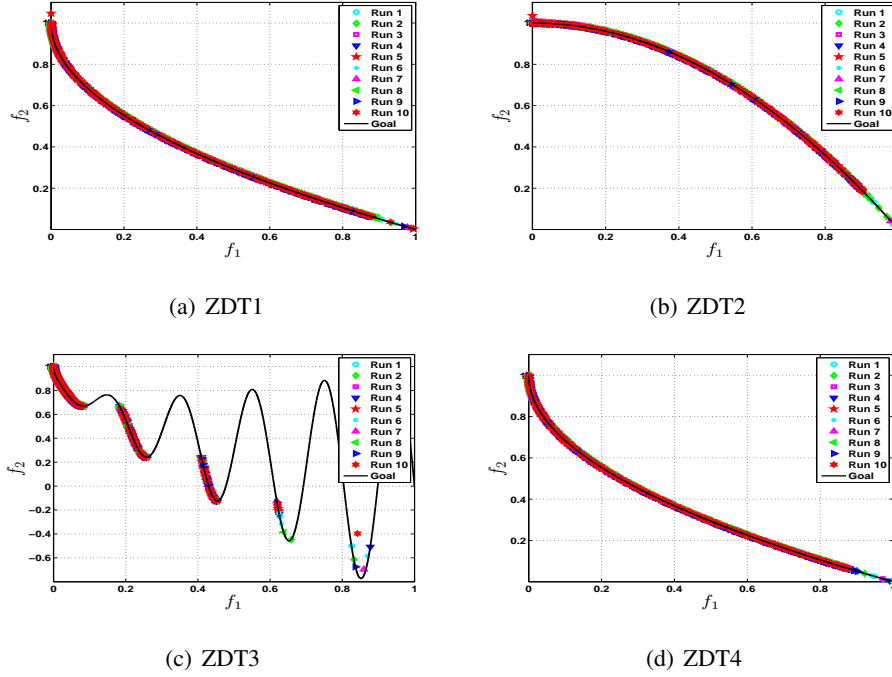


(c) ZDT3



(d) ZDT4

Figure 2.7: ZDT functions with 5 variables for NSGA-II_{SBX}.

Figure 2.8: ZDT functions with 5 variables for NSGA-II_{DE}.

general, all the methods exhibit a number of solutions that dominate to the other methods, except for ZDT4 when MOPSO is compared with MOEAD_{SBX} and MOEAD_{DE}. For all the ZDT functions, NSGA-II_{DE} exhibits the grater domination average over the other methods, however for ZDT4, MOPSO dominates to NSGA-II_{SBX} and NSGA-II_{DE} and does not dominate or MOEAD_{SBX} or MOEAD_{DE}. This fact, adds reliability to MOEAD over the other methods. Regarding to SBX and DE, it is possible to highlight how for all the experiments NSGA-II and MOEAD improved the dominance rate when both used DE.

2.6 Summary

This chapter showed the usefulness of the EAs in the multi-objective optimization. The EAs are able to handle several variables, non-linear problems and constraints. With the provided examples, sometimes an EA can be better than other one for some problems, but there is not one that dominates to the others completely. The recombination operators exhibit the same issue. Although the EAs showed success to optimize mathematical functions, it is necessary to

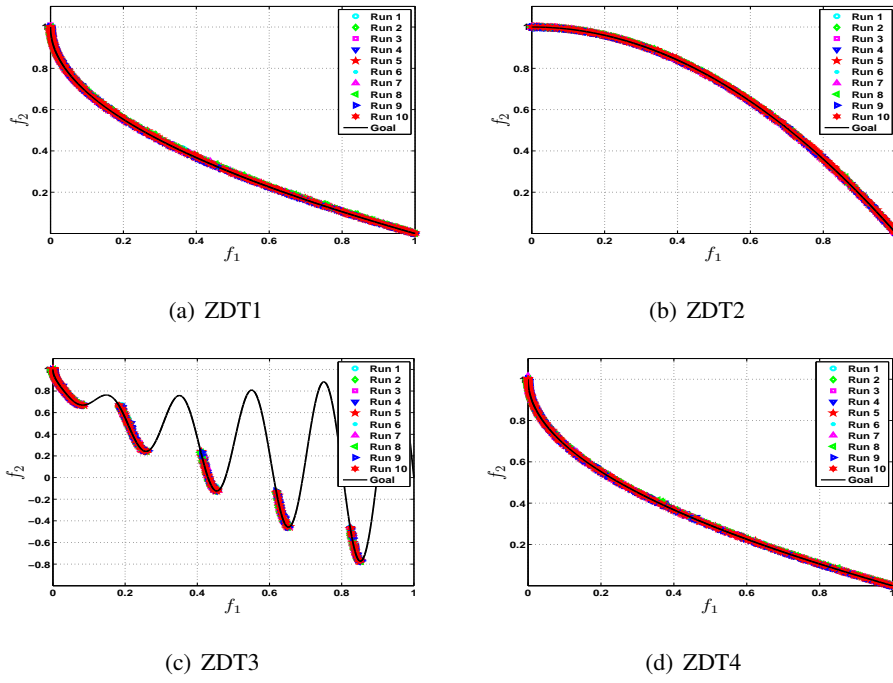


Figure 2.9: ZDT functions with 5 variables for MOEAD_{SBX}.

explore their behavior when analog circuits are optimized, then the next chapter is devoted to using the EAs but this time for circuit optimization.

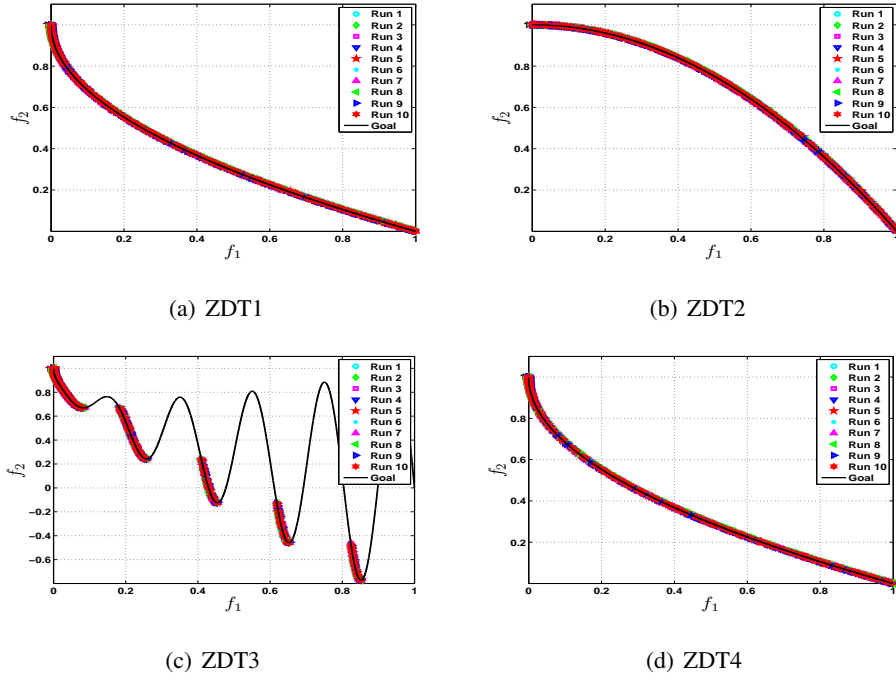
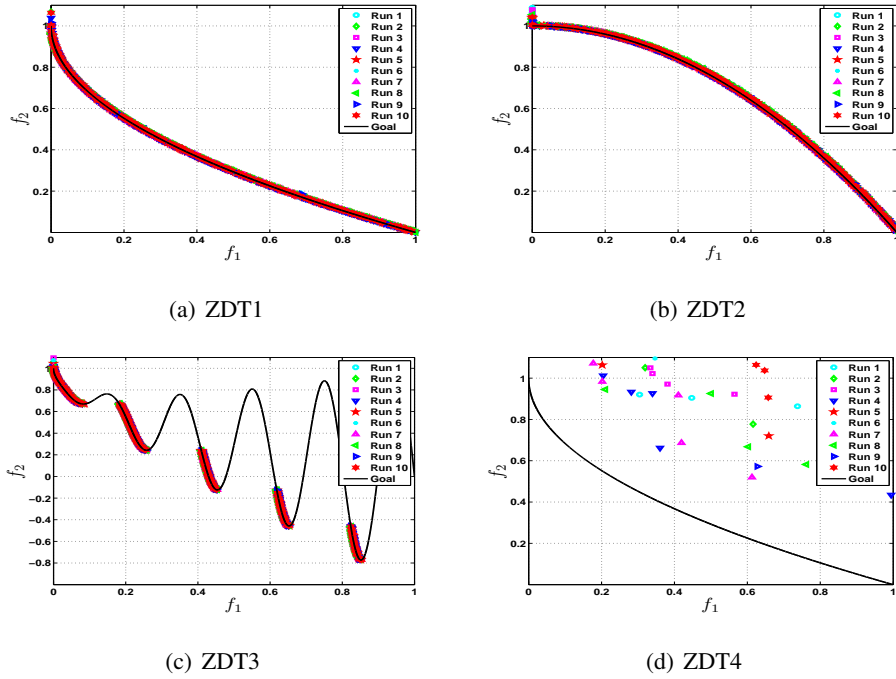
Figure 2.10: ZDT functions with 5 variables for MOEAD_{DE}.

Figure 2.11: ZDT functions with 5 variables for MOPSO.

Table 2.2: Coverage metric for each method for ZDT functions with 5 variables

Function	Method	NSGA-II _{SBX}	NSGA-II _{DE}	MOEAD _{SBX}	MOEAD _{DE}	MOPSO
ZDT1	NSGA-II _{SBX}	-	0.99207	0.99663	0.99573	0.99837
	NSGA-II _{DE}	0.99557	-	0.9982	0.99693	1
	MOEAD _{SBX}	0.66554	0.65982	-	0.70411	0.84746
	MOEAD _{DE}	0.85573	0.8487	0.9405	-	0.97983
	MOPSO	0.51837	0.50877	0.7202	0.62313	-
ZDT2	NSGA-II _{SBX}	-	0.9903	0.99257	0.9925	0.9973
	NSGA-II _{DE}	0.9952	-	0.9936	0.9924	1
	MOEAD _{SBX}	0.88107	0.88209	-	0.88554	0.98769
	MOEAD _{DE}	0.93807	0.9379	0.9518	-	0.99883
	MOPSO	0.28192	0.28992	0.24402	0.21074	-
ZDT3	NSGA-II _{SBX}	-	0.9885	0.9958	0.99507	0.99833
	NSGA-II _{DE}	0.99143	-	0.99687	0.9949	0.9977
	MOEAD _{SBX}	0.6112	0.58085	-	0.60574	0.78931
	MOEAD _{DE}	0.7335	0.7405	0.91927	-	0.94647
	MOPSO	0.6393	0.5812	0.74777	0.6114	-
ZDT4	NSGA-II _{SBX}	-	0.99507	0.99623	0.99577	0.99737
	NSGA-II _{DE}	0.9984	-	0.99883	0.9969	1
	MOEAD _{SBX}	0.8948	0.88544	-	0.8814	1
	MOEAD _{DE}	0.97103	0.97087	0.9782	-	1
	MOPSO	0.55015	0.076391	0	0	-

Chapter 3

Circuit Optimization

3.1 Introduction

This chapter is devoted to show the optimization process of analog integrated circuits by using evolutionary algorithms. In the first section, the proposed optimization methodology framework based on successive simulations linking a circuit simulator, is presented.

Next, it is analyzed the performance of NSGA-II and MOEAD with two different recombination operators: SBX and DE, through the optimization of two mixed-mode circuits each one with different number of design variables and three objective functions. The chapter ends showing the optimization of nine amplifiers all of them with different number of design variables and eleven objective functions.

3.2 Optimization Methodology Framework

To perform circuit optimization, after an initialization procedure, it is necessary in each generation to evaluate the population by linking a circuit simulator and by modifying each design variable (as transistor width, transistor length or bias sources), and collecting these results. Next, the new population is generated from the best individuals (or non-dominated solutions). This process continues until the current generation reaches the maximum number of generations as shown in Fig. 3.1.

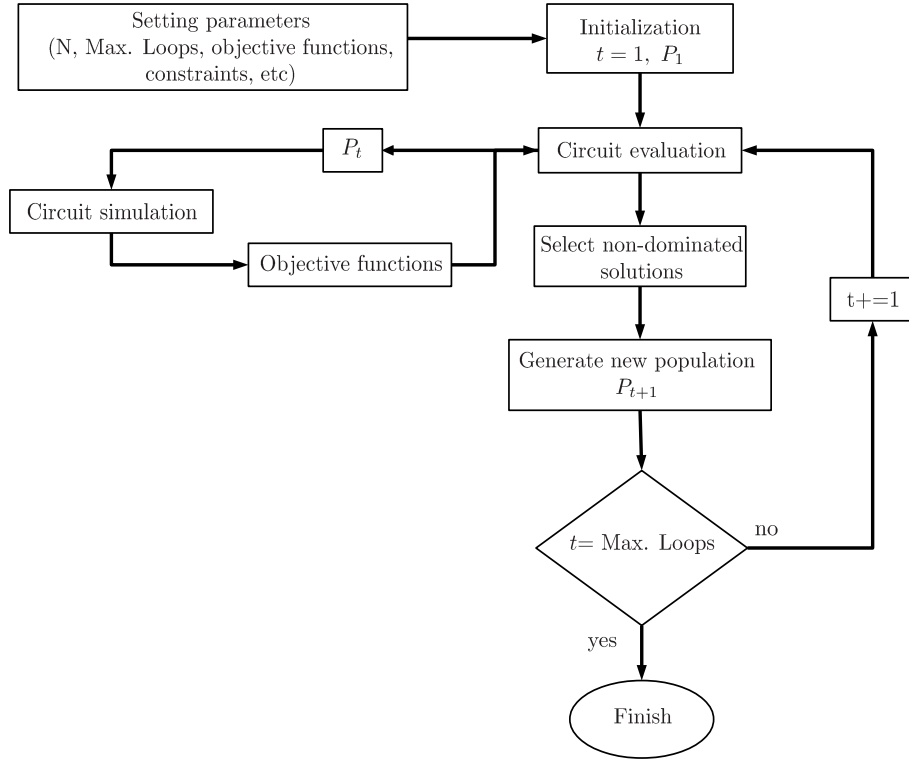


Figure 3.1: Optimization Methodology Framework

In this work, the optimization is performed by linking HSPICE as circuit simulator in order to compute the circuit performances. The simulation results are gathered by using the HSPICE “.MEAS” [81] instruction that is capable to save electronic measurements in the output listing. This instruction allows making measurements from the saturation condition of a specific transistor (through its drain current and voltage in its terminals) and also allows making measurements on a specific output analysis such as AC or DC.

Table 3.1 shows a library example that includes specific measurements for AC or DC circuit analysis. From an AC analysis, it is possible to use a .MEAS sentence to calculate the voltage gain (in dB units) at 1Hz in the OUT circuit node and save this value in the variable “av”. In the same way, it is possible to measure the band width signal when the variable av has decreased 3 dB and save this value in the variable “bw”. Finally, it is called a DC analysis to perform the voltage offset measurement by saving the input voltage value in the variable “offs” when the output voltage is zero.

Table 3.1: Measurements Library Example

.LIB MEASLIB	* Library name
.AC dec 1 100 1G	* Execute an AC Analysis
.MEAS AC av MAX Vdb(OUT) FROM=1 TO=1	* Calculating Gain in db
.MEAS AC bw TRIG Vdb(OUT) AT=1 TARG Vdb(OUT) VAL='av-3' CROSS=1	* Calculating f_{-3db}
.DC VIN 1.5 -1.5 .01	* Execute a DC Analysis
.MEAS DC offs FIND V(OUT) WHEN V(IN)=0 CROSS=1	* Calculating offset
.ENDL MEASLIB	

3.3 Circuit Optimization with SBX and DE

This section is devoted to show the optimization of two analog circuits by using DE and SBX as recombination operators. Figure 3.2(a) depicts a Positive-type Second Generation Current Conveyor (CCII⁺) [82] which accomplishes $V_X = V_Y$ and $I_Z = I_X$. Next, Fig. 3.2(b) depicts a Negative-type Second Generation Current Conveyor (CCII⁻) [83] which accomplishes $V_X = V_Y$ and $I_Z = -I_X$. Both circuits are biased with $I_{ref} = 50\mu A$, $-V_{ss} = V_{dd} = 1.5V$, and it is assumed that all the MOSFETs have the same transistor length ($0.7\mu m$), then the transistor sizing is made by varying the transistor widths from $0.35\mu m$ to $100\mu m$. For each electrical measurement there is a load capacitor of 1pF and the SPICE simulations are performed with a LEVEL 49 standard CMOS Technology of $0.35\mu m$. Table 3.2 shows the codification of W of the transistors for CCII⁺ ($n = 2$) and CCII⁻ ($n = 8$).

3.3.1 Multi-Objective Optimization Problem Formulation

The objectives to optimize are: gain, offset and band width (BW) for voltage and current mode. Gain is the relation between the Y to X voltage transfer, and the X to Z current transfer. Offset is a voltage or current value between Y-X or X-Z, and BW is always expressed in Hertz. For voltage and current optimization of these circuits is desired a gain closer to unity, a minimum offset and a maximum BW.

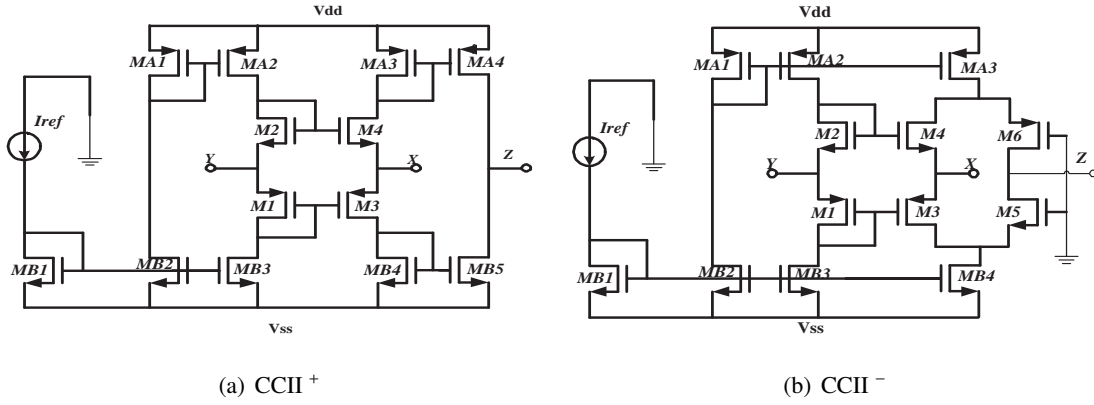


Figure 3.2: Tested Circuits.

Table 3.2: Variables encoding for the CCII's

Variable Name	Transistors	
	CCII ⁺	CCII ⁻
W_1	$MA_1, \dots, MA_4, M_1, M_3$	MB_1, MB_2, MB_3
W_2	$MB_1, \dots, MB_5, M_2, M_4$	MB_4
W_3	-	MA_1, MA_2
W_4	-	MA_3
W_5	-	M_1, M_3
W_6	-	M_2, M_4
W_7	-	M_5
W_8	-	M_6

For both CCII's the optimization problem is expressed as:

$$\begin{aligned}
 &\text{minimize } \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x})]^T \\
 &\text{subject to } h_l(\mathbf{x}) \geq 0, \quad l = 1 \dots p, \\
 &\text{where } \mathbf{x} \in X.
 \end{aligned} \tag{3.1}$$

$\mathbf{f}(\mathbf{x})$ is the vector formed by three objectives:

- $f_1(\mathbf{x}) = |1 - \text{Gain}|$.
- $f_2(\mathbf{x}) = -1 * \text{BW}$.
- $f_3(\mathbf{x}) = \text{Offset}$.

where $X : \mathbb{R}^n \mid 0.35 \mu m \leq W_i \leq 100 \mu m$ is the decision space (X) for the n variables. Finally, $h_l(\mathbf{x})$, $l = 1 \dots p$ are performance constraints, in our experiments we include the next constraints:

- The saturation condition in all transistors.
- $|1-\text{Gain}| < 0.1$.
- $\text{BW} > 100\text{MHz}$.
- Voltage offset $< 1\text{mV}$.
- Current offset $< 1\mu\text{A}$.

3.3.2 Results

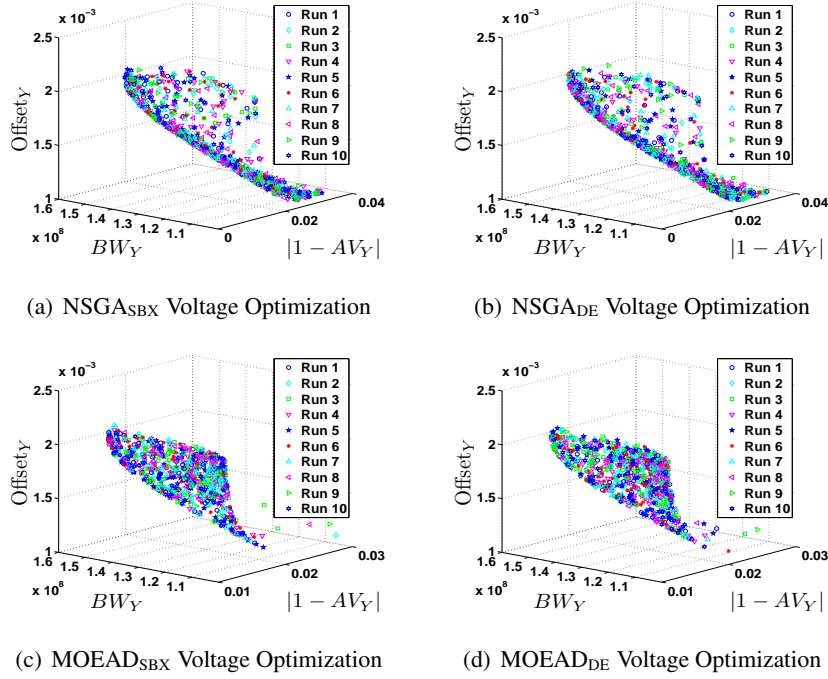
The circuits are optimized along 10 runs (in a dual processor 2GHz, RAM 2GHz). For CCII^+ it was selected $N=80$ ($H = 12$ for MOEAD) along 80 generations. For CCII^- it was selected $N=152$ ($H = 17$ for MOEAD) along 200 generations.

Figures 3.3 - 3.4 depict the last non-dominated solutions for voltage and current optimization for the NSGA-II and MOEAD approaches for the CCII^+ along 10 runs, herein the Pareto fronts of these experiments are similar.

Tables 3.3 and 3.4 show the worst, mean and best objective values found by each method as result of the optimization of both circuits. Among all these results it is bear out that the constraints of gain, band width and offset were accomplished in voltage and current mode. The loop time is comparable for all the experiments, only MOEAD_{DE} particularly preserves always the less mean time.

Regarding to the voltage optimization for the CCII^+ , both NSGA-II experiments achieved similar objective values between NSGA-II_{SBX} and NSGA-II_{DE} . Both MOEAD experiments also have similar objective values between MOEAD_{SBX} and MOEAD_{DE} . The average difference among all the experiments for the voltage optimization of CCII^+ is 3% (left side of Table 3.3).

For the current optimization of the CCII^+ , the NSGA-II experiments found the same best, worst and mean objective values and the MOEAD experiments have similar objective values.

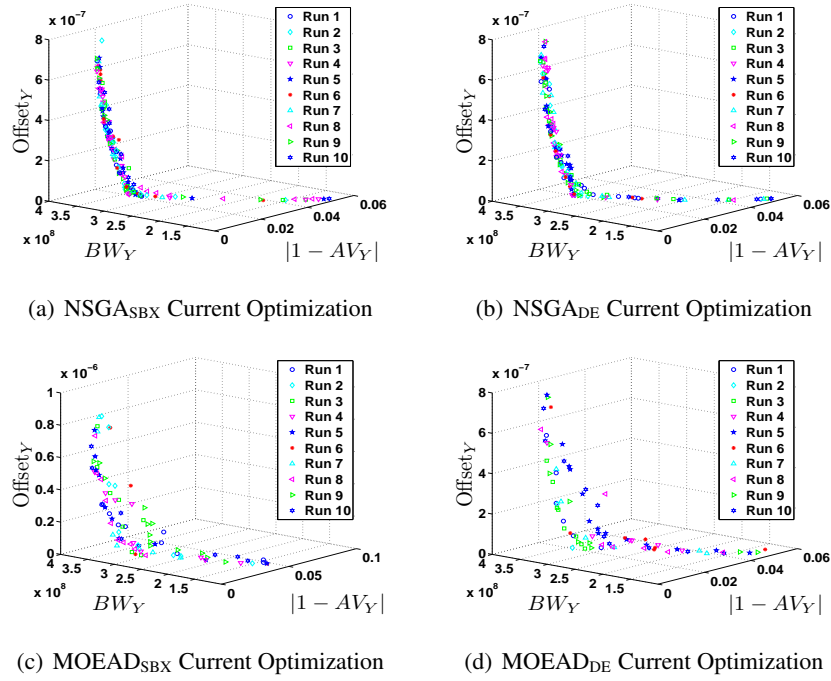
Figure 3.3: CCII⁺ Voltage Optimization.

In this case, the average difference among all the results of the experiments is 17%, however the average difference among the mean values is 6% (left side of Table 3.4).

The voltage optimization for the CCII⁻ does not have the same similitude as the CCII⁺ among the experiments, however the average difference among the objective values is 15% (right side of Table 3.3). The current optimization of the CCII⁻ exhibits an average difference among the experiments of 24% (right side of Table 3.4). These higher average difference values of CCII⁻ compared with CCII⁺ can be attributed to the increment of the population size however the constraints are accomplished.

Table 3.5 displays the coverage metric which shows how much the methods in columns are dominated by the methods in rows. These values are calculated from gathering all the non dominated solutions of each run for each experiment. All the values are great than zero, this denotes that for all the experiments there exist solutions that dominate solutions to other methods, in other words: no method dominates completely to another method.

For the voltage optimization of the CCII⁺, MOEAD exhibits the higher dominance percent-

Figure 3.4: CCII⁺ Current Optimization.

age over the NSGA-II methods, for instance, while NSGA-II_{SBX} dominates to MOEAD_{SBX} a 0.315 value, MOEAD_{SBX} dominates to NSGA-II_{SBX} a 0.41413 value. This behavior is repeated by comparing NSGA-II_{SBX} with MOEAD_{DE} (0.302 vs. 0.41413), NSGA-II_{DE} with MOEAD_{SBX} (0.302 vs. 0.41738) and NSGA-II_{DE} with MOEAD_{DE} (0.30112 vs. 0.43013). Unlike the voltage optimization of the CCII⁺, its current optimization exhibits the opposite behavior: NSGA-II methods exhibit the higher dominance percentage over MOEAD.

The voltage optimization of the CCII⁻ exhibits that methods with SBX have higher dominance percentage than methods with DE. The current optimization of the CCII⁻ shows that NSGA-II methods dominates to MOEAD methods, for instance NSGA-II_{SBX} dominates to MOEAD_{SBX} a 0.81811 value that is higher than the 0.43067 value that MOEAD_{SBX} dominates to NSGA-II_{SBX}. Such behavior is the same for all the comparisons between NSGA-II and MOEAD methods in the current optimization.

Table 3.3: CCII's Voltage Optimization Results

	CCII ⁺				CCII ⁻			
	$n = 2, N = 80, M = 3$				$n = 6, N = 152, M = 3$			
	Loop time (secs)	Gain V/V	BW Hz	Offset Volts	Loop time (secs)	Gain V/V	BW Hz	Offset Volts
	NSGA _{SBX}							
WORST	9	0.969	1.000E8	2.254E-3	17	0.981	1.261E8	4.321E-3
MEAN	8.4	0.983	1.240E8	1.591E-3	16.2	0.986	1.443E8	2.532E-3
BEST	8	0.989	1.548E8	1.073E-3	16	0.988	1.635E8	9.673E-4
	NSGA _{DE}							
WORST	9	0.968	1.000E8	2.247E-3	17	0.941	1.296E8	4.468E-3
MEAN	8.4	0.983	1.244E8	1.595E-3	16.7	0.983	1.5074E8	2.407E-3
BEST	8	0.989	1.548E8	1.072E-3	16	0.988	1.599E8	3.377E-4
	MOEAD _{SBX}							
WORST	9	0.971	1.000E8	2.187E-3	17	0.980	1.138E8	3.693E-3
MEAN	8.5	0.987	1.195E8	1.866E-3	16.3	0.986	1.384E8	1.824E-3
BEST	8	0.989	1.547E8	1.120E-3	16	0.988	1.560E8	3.322E-4
	MOEAD _{DE}							
WORST	9	0.973	1.001E8	2.195E-3	17	0.985	1.370E8	3.288E-3
MEAN	8.39	0.987	1.189E8	1.869E-3	16.2	0.987	1.516E8	2.130E-3
BEST	8	0.989	1.546E8	1.115E-3	16	0.989	1.610E8	5.339E-4

Table 3.4: CCII's Current Optimization Results

	CCII ⁺				CCII ⁻			
	$n = 2, N = 80, M = 3$				$n = 6, N = 152, M = 3$			
	Loop time (secs)	Gain $ I/I $	BW Hz	Offset Amp.	Loop time (secs)	Gain $ I/I $	BW Hz	Offset Amp.
	NSGA _{SBX}							
WORST	8	0.948	1.126E8	7.501E-7	15	0.913	1.808E8	9.737E-7
MEAN	7.3	0.978	3.423E8	2.277E-7	14.2	0.982	1.837E8	2.652E-7
BEST	7	0.982	3.903E8	2.269e-011	14	0.992	3.361E8	1.831E-9
	NSGA _{DE}							
WORST	8	0.948	1.138E8	7.451E-7	15	0.962	1.187E8	9.523E-7
MEAN	7.5	0.978	3.342E8	2.241E-7	14.2	0.984	2.030E8	4.424E-7
BEST	7	0.982	3.903E8	1.522e-012	14	0.991	3.534E8	1.453E-8
	MOEAD _{SBX}							
WORST	8	0.953	1.335E8	8.304E-7	15	0.958	1.088E8	9.470E-7
MEAN	7.3	0.975	3.092E8	2.338E-7	14.1	0.984	1.953E8	3.907E-7
BEST	7	0.982	3.891E8	9.868e-010	14	0.993	3.222E8	4.280E-8
	MOEAD _{DE}							
WORST	8	0.949	1.189E8	7.441E-7	15	0.978	1.167E8	9.195E-7
MEAN	7.2	0.974	2.980E8	1.755E-7	14.1	0.985	1.710E8	3.838E-7
BEST	7	0.981	3.893E8	8.624e-011	14	0.989	2.604E8	6.291E-8

Table 3.5: Coverage Metric for CCII's Current and Voltage Optimization

	NSGA _{SBX}	NSGA _{DE}	MOEAD _{SBX}	MOEAD _{DE}
Voltage Optimization CCII ⁺				
NSGA _{SBX}	-	0.14663	0.315	0.302
NSGA _{DE}	0.19763	-	0.3085	0.30112
MOEAD _{SBX}	0.41413	0.40525	-	0.22787
MOEAD _{DE}	0.41738	0.43013	0.22125	-
Current Optimization CCII ⁺				
NSGA _{SBX}	-	0.7902	0.98331	0.9521
NSGA _{DE}	0.65499	-	0.96369	0.93925
MOEAD _{SBX}	0.62166	0.65007	-	0.85938
MOEAD _{DE}	0.71781	0.74556	0.91847	-
Voltage Optimization CCII ⁻				
NSGA _{SBX}	-	0.87	0.92	0.81333
NSGA _{DE}	0.76	-	0.73867	0.88
MOEAD _{SBX}	0.94667	0.90667	-	0.94667
MOEAD _{DE}	0.78	0.82	0.56	-
Current Optimization CCII ⁻				
NSGA _{SBX}	-	0.7439	0.81811	0.92998
NSGA _{DE}	0.36855	-	0.69242	0.86131
MOEAD _{SBX}	0.43067	0.604	-	0.78778
MOEAD _{DE}	0.59778	0.51	0.79111	-

3.4 Circuit Optimization with NSGA-II, MOEAD and MOPSO

In this section there are optimized with NSGA-II, MOEA/D and MOPSO, nine Current Feed-back Operational Amplifiers (CFOAs) all of them designed from the three Voltage Followers (VFs) depicted in Fig. 3.5. Each CFOA consists of two VFs and a simple current mirror to bind them. There are nine different combinations labeled as: CFOA_{AA}, CFOA_{AB}, ..., CFOA_{CC} depicted in Fig. 3.6. Regarding to the voltage or current gain among the different ports of a CFOA, it is desired a gain closer to the unity and a high frequency for the bandwidth. Also it is desired a high resistance on ports *Y* and *Z*, and a low resistance on ports *X* and *W*.

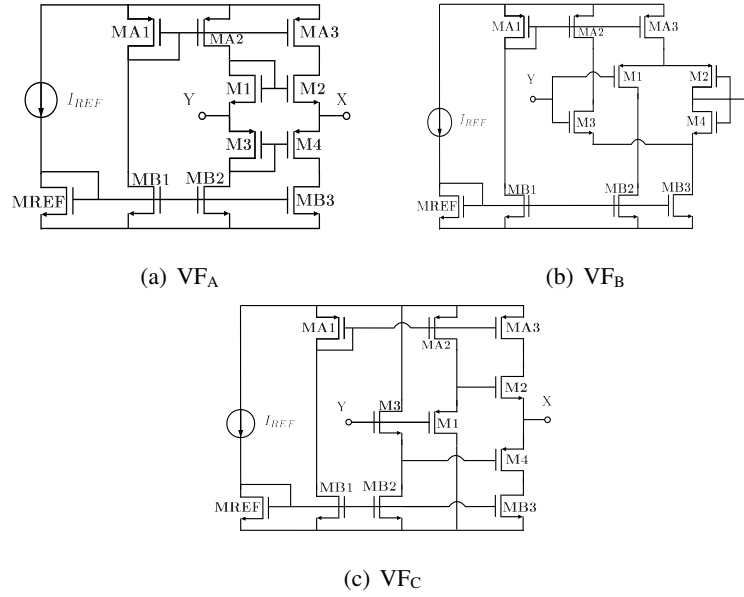


Figure 3.5: Three different Voltage Followers.

The optimization systems based in NSGA-II, MOEAD and MOPSO, are performed with a PERL script and the circuit simulations are made with a SPICE simulator by modifying each transistor width (W_i) and length (L).

The CFOAs are biased with $VDD = 1.5V$ and $VSS = -1.5V$. The electrical measurements were executed with a load capacitor of 1pF and the SPICE simulations were performed with a LEVEL 49 standard CMOS Technology of $0.18 \mu m$.

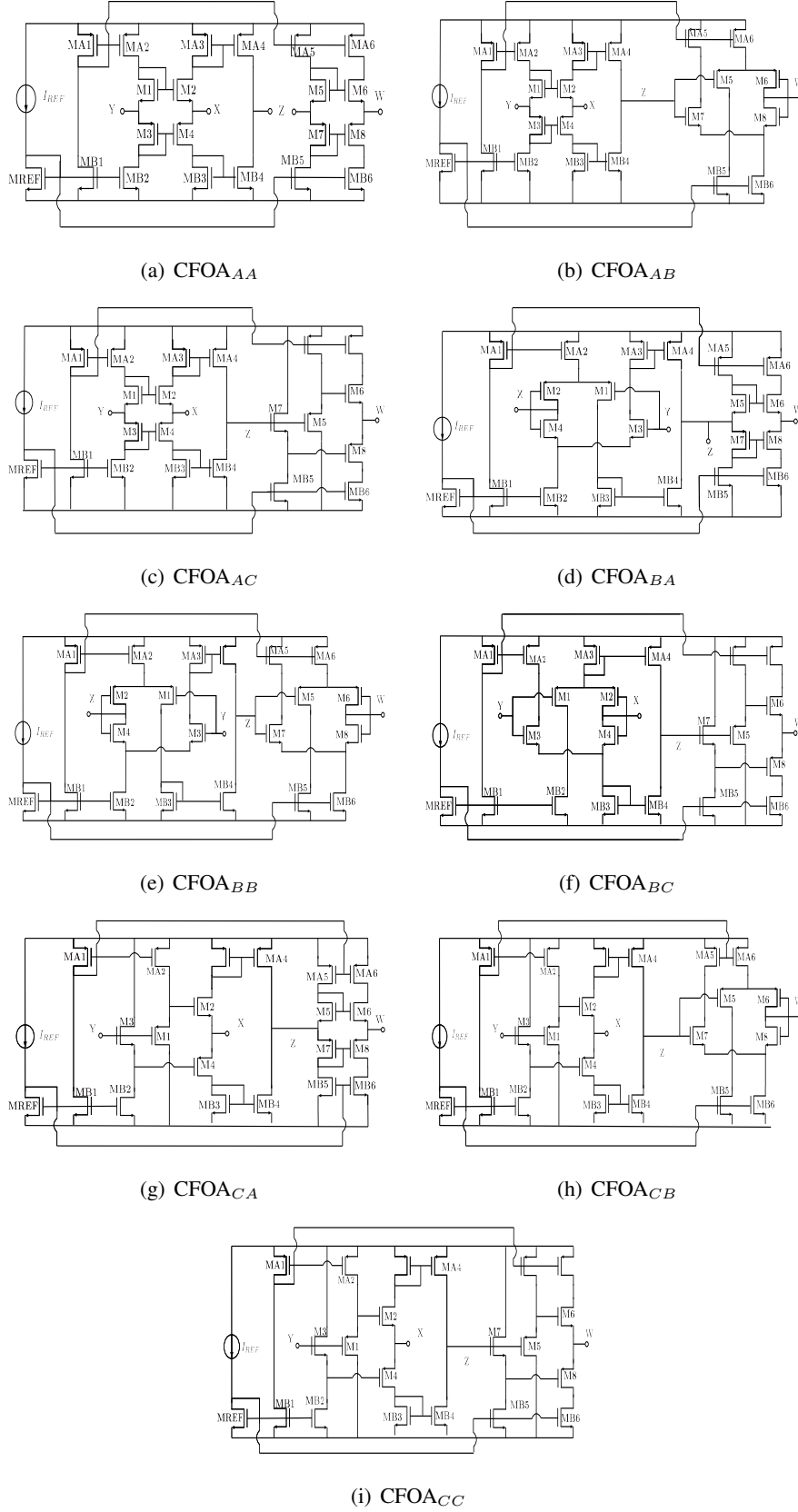


Figure 3.6: CFOAs.

All CFOAs are encoded with transistors lengths (L_i) and widths (W_i), where i represents a specific transistor (or transistors which share the same length or/and width) of the circuit. The decision space for L is $0.18\mu m \leq L \leq 0.9\mu m$, for W is $0.18\mu m \leq W \leq 200\mu m$. All L and W values are rounded to multiples of the minimum allowed by the technology process, because besides the simulator can not handle continuous values, the scaling process to other technologies can be done easily. Besides L s and W s, there exists one more variable for I_{REF} which decision space is $10\mu A \leq I_{REF} \leq 400\mu A$. The variables encoding for each CFOA are listed in Tables 3.6 to 3.14, where are the variables assigned for the different transistors and the current variable I_{REF} .

3.4.1 Multi-Objective Optimization Problem Formulation

For all CFOAs the optimization problem is expressed as:

$$\begin{aligned} & \text{minimize } \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_{11}(\mathbf{x})]^T \\ & \text{subject to } h_l(\mathbf{x}) \geq 0, \quad l = 1 \dots p, \\ & \text{where } \mathbf{x} \in X. \end{aligned} \tag{3.2}$$

$\mathbf{f}(\mathbf{x})$ is the vector formed by eleven objectives:

- $f_1(\mathbf{x})$ = Power consumption .
- $f_2(\mathbf{x})$ = |1 - Voltage gain from Y port to X port| ($GAIN_X$).
- $f_3(\mathbf{x})$ = -1* Voltage band width from Y port to X port (BW_X).
- $f_4(\mathbf{x})$ = -1* Input resistance on Y port (Z_Y).
- $f_5(\mathbf{x})$ = Output resistance on X port (Z_X).
- $f_6(\mathbf{x})$ = |1 - Current gain from X port to Z port| ($GAIN_Z$).
- $f_7(\mathbf{x})$ = -1* Current band width from X port to Z port (BW_Z).
- $f_8(\mathbf{x})$ = -1* Output resistance on Z port (Z_Z).
- $f_9(\mathbf{x})$ = |1 - Voltage gain from Z port to W port| ($GAIN_W$).

- $f_{10}(\mathbf{x}) = -1 \cdot \text{Voltage band width from Z port to W port (BW}_W\text{)}.$
- $f_{11}(\mathbf{x}) = \text{Output resistance on W port (Z}_W\text{)}.$

where $X : \mathbb{R}^n \mid 0.18 \mu m \leq L_i \leq 0.9 \mu m, 0.18 \mu m \leq W_j \leq 200 \mu m, 10 \mu m \leq I_{BIAS} \leq 400 \mu A,$ is the decision space for the n variables. Finally, $h_l(\mathbf{x}), l = 1 \dots p$ are performance constraints, in our experiments we include the saturation condition in all transistors as constraints.

Table 3.6: CFOA_{AA} encoding

<i>gene</i>	Design Variable	<i>Encoding</i>
x_1	L	All transistors
x_2	W_1	MREF, MB1-MB4, M1, M2
x_3	W_2	MA1-MA4, M3, M4
x_4	W_3	MB5, MB6, M5, M6
x_5	W_4	MA5, MA6, M7, M8
x_6	I	I_{REF}

Table 3.8: CFOA_{AC} encoding

<i>gene</i>	Design Variable	<i>Encoding</i>
x_1	L_1	MREF, MA1-MA4, MB1-MB4, M1-M4
x_2	L_2	MA5, MA6, MB5, MB6, M6-M8
x_3	W_1	MREF, MB1-MB4, M1, M2
x_4	W_2	MA1-MA4, M3, M4
x_5	W_3	MB5, MB6
x_6	W_4	MA5, MA6
x_7	W_5	M5
x_8	W_6	M6
x_9	W_7	M7
x_{10}	W_8	M8
x_{11}	I	I_{REF}

Table 3.7: CFOA_{AB} encoding

<i>gene</i>	Design Variable	<i>Encoding</i>
x_1	L_1	MREF, MA1-MA4, MB1-MB4, M1-M4
x_2	L_2	MA5, MA6, MB5, MB6, M6-M8
x_3	W_1	MREF, MB1-MB4, M1, M2
x_4	W_2	MA1-MA4, M3, M4
x_5	W_3 $0.5 \cdot W_3$	MB6, M5, M6 MB5
x_6	W_4 $0.5 \cdot W_4$	MA6, M7, M8 MA5
x_7	I	I_{REF}

Table 3.9: CFOA_{BA} encoding

<i>gene</i>	Design Variable	<i>Encoding</i>
x_1	L_1	MREF, MA1-MA4, MB1-MB4, M1-M4
x_2	L_2	MA5, MA6, MB5, MB6, M6-M8
x_3	W_1	MREF, MB1, MB2, M3, M4
x_4	W_2	MA1, MA2, M1, M2
x_5	W_3	MB3, MB4
x_6	W_4	MA3, MA4
x_7	W_5	MB5, MB6, M5, M6
x_8	W_6	MA5, MA6, M7, M8
x_9	I	I_{REF}

Table 3.10: CFOA_{BB} encoding

<i>gene</i>	Design Variable	Encoding
x_1	L_1	MREF, MA1-MA4, MB1-MB4, M1-M4
x_2	L_2	MA5, MA6, MB5, MB6, M6-M8
x_3	W_1	MREF, MB1, MB2, M3, M4
x_4	W_2	MA1, MA2, M1, M2
x_5	W_3	MB3, MB4
x_6	W_4	MA3, MA4
x_7	W_5	MB6, M7, M8
	$0.5 \cdot W_5$	MB5
x_8	W_6	MA6, M5, M6
	$0.5 \cdot W_6$	MA5
x_9	I	I_{REF}

Table 3.11: CFOA_{BC} encoding

<i>gene</i>	Design Variable	Encoding
x_1	L_1	MREF, MA1-MA4, MB1-MB4, M1-M4
x_2	L_2	MA5, MA6, MB5, MB6, M6-M8
x_3	W_1	MREF, MB1, MB2, M3, M4
x_4	W_2	MA1, MA2, M1, M2
x_5	W_3	MB3, MB4
x_6	W_4	MA3, MA4
x_7	W_5	MB5, MB6
x_8	W_6	MA5, MA6
x_9	W_7	M5
x_{10}	W_8	M6
x_{11}	W_9	M7
x_{12}	W_{10}	M8
x_{13}	I	I_{REF}

Table 3.12: CFOA_{CA} encoding

<i>gene</i>	Design Variable	Encoding
x_1	L	All transistors
x_2	W_1	MREF, MB1-MB4
x_3	W_2	MA1-MA4
x_4	W_3	M3
x_5	W_4	M1
x_6	W_5	M2
x_7	W_6	M4
x_8	W_7	MB5, MB6, M5, M6
x_9	W_8	MA5, MA6, M7, M8
x_{10}	I	I_{REF}

Table 3.13: CFOA_{CB} encoding

<i>gene</i>	Design Variable	Encoding
x_1	L_1	MREF, MA1-MA4, MB1-MB4, M1-M4
x_2	L_2	MA5, MA6, MB5, MB6, M6-M8
x_3	W_1	MREF, MB1-MB4
x_4	W_2	MA1-MA4
x_5	W_3	M3
x_6	W_4	M1
x_7	W_5	M2
x_8	W_6	M4
x_9	W_7	MB6, M7, M8
	$0.5 \cdot W_3$	MB5
x_{10}	W_8	MA6, M5, M6
	$0.5 \cdot W_4$	MA5
x_{11}	I	I_{REF}

Table 3.14: CFOA_{CC} encoding

<i>gene</i>	Design Variable	<i>Encoding</i>
x_1	L_1	MREF, MA1-MA4, MB1-MB4, M1-M4
x_2	L_2	MA5, MA6, MB5, MB6, M6-M8
x_3	W_1	MREF, MB1-MB4
x_4	W_2	MA1-MA4
x_5	W_3	M3
x_6	W_4	M1
x_7	W_5	M2
x_8	W_6	M4
x_9	W_7	MB5, MB6
x_{10}	W_8	MA5, MA6
x_{11}	W_9	M6
x_{12}	W_{10}	M5
x_{13}	W_{11}	M7
x_{14}	W_{12}	M8
x_{15}	I	I_{REF}

3.4.2 Results

The optimization for the CFOAs was performed with a population size of 600 along 200 generations. Tables 3.15-3.17 show the worst, best and mean values from the optimization results of the CFOAs. The fact that this optimization problem has a lot of objective functions makes difficult finding solutions that dominates to the others solutions from the same method or indeed from the another methods, then there is no dominance percentage of any method over another one.

For the CFOA_{AA} (upper section of Table 3.15), MOEAD accomplishes the major best objectives except for BW_Z (that exhibits 6% of difference compared with the best value of

NSGA-II) and $GAIN_W$ (that exhibits less than 1% compared with the best value of NSGA-II). With the $CFOA_{AB}$ optimization (mid section of Table 3.15), MOEAD achieved the best values of each objective function compared with the other methods. In the optimization of the $CFOA_{AC}$ (lower section of the Table 3.15), again MOEAD has the best objective values except for power consumption (with a difference of 15% compared with the best NSGA-II value) and $GAIN_W$ (with a difference less than 1% compared with the best MOPSO value).

In the $CFOA_{BA}$ optimization (upper section of Table 3.16), both NSGA-II and MOEAD has the best vales, however, despite MOEAD achieved one best value than NSGA-II, this last exhibits the best value for the power consumption. The optimization of $CFOA_{BB}$ (mid section of Table 3.16) is led by MOEAD, but different to the previous circuits, PSO achieved best values than NSGA-II. With the $CFOA_{BC}$ (lower section of Table 3.16), MOEAD exhibit the best objective values, followed by NSGA-II and sometimes MOPSO provides better values than NSGA-II (in Z_X , BW_Z , BW_W and Z_W).

The optimization of the $CFOA_{CA}$, $CFOA_{CB}$ and $CFOA_{CC}$ (Table 3.17) is completely led by MOEAD followed by NSGA-II and finally by MOPSO.

Table 3.15: Results of optimization for CFOA_{AA}, CFOA_{AB} and CFOA_{AC}

CFOA	Method		POWER Watts	GAIN _X $ \frac{V_X}{V_Y} $	BW _X Hz	Z _Y Ω	Z _X Ω	GAIN _Z $ \frac{V_Z}{V_X} $	BW _Z Hz	Z _Z Ω	GAIN _W $ \frac{V_X}{V_Y} $	BW _W Hz	Z _W Ω
CFOA _{AA}	NSGA-II _{DE}	WORST	1.995E-3	0.972	4.597E7	1.776E3	3.439E3	0.970	5.227E7	528.800	0.970	9.127E5	1.650E5
		MEAN	6.634E-4	0.983	1.663E8	9.247E3	905.255	0.985	2.876E8	4.411E4	0.980	1.317E8	9.097E3
		BEST	1.613E-4	0.988	4.077E8	7.432E4	286.030	1.000	8.102E8	3.914E5	0.989	4.428E8	207.250
	MOEAD _{DE}	WORST	4.548E-3	0.980	4.957E7	683.530	2.332E3	0.970	3.841E7	72.721	0.976	1.276E6	1.229E5
		MEAN	2.676E-3	0.981	3.935E8	5.949E5	968.437	0.974	5.442E8	7.470E4	0.979	4.495E8	1.004E4
		BEST	1.577E-4	0.988	6.596E8	3.473E6	135.930	1.000	7.566E8	4.425E5	0.988	7.086E8	50.128
	MOPSO	WORST	1.222E-3	0.985	8.393E7	1.481E3	952.540	0.970	5.692E7	1.786E3	0.980	3.042E7	3.947E3
		MEAN	7.740E-4	0.988	1.426E8	3.566E3	527.923	0.978	9.459E7	6.074E3	0.985	8.921E7	1.313E3
		BEST	3.931E-4	0.988	2.397E8	7.822E3	341.670	0.997	2.265E8	3.649E4	0.989	2.106E8	403.630
CFOA _{AB}	NSGA-II _{DE}	WORST	1.658E-3	0.980	9.465E7	3.752E3	1.222E3	0.970	8.054E7	4.760E3	0.970	1.225E6	1.180E5
		MEAN	6.168E-4	0.983	1.808E8	1.169E4	725.639	0.983	2.878E8	2.555E4	0.976	3.482E7	1.005E4
		BEST	3.323E-4	0.988	4.142E8	3.048E4	269.140	0.999	6.075E8	1.746E5	0.980	8.528E7	1.677E3
	MOEAD _{DE}	WORST	3.808E-3	0.980	5.814E7	1.256E3	2.417E3	0.970	6.376E7	1.324E3	0.970	7.871E5	1.897E5
		MEAN	1.708E-3	0.982	2.868E8	3.248E4	1.111E3	0.977	5.025E8	8.404E4	0.973	1.194E8	2.439E4
		BEST	1.575E-4	0.988	6.533E8	1.155E5	131.430	1.000	7.587E8	4.794E5	0.981	2.161E8	481.750
	MOPSO	WORST	1.024E-3	0.982	9.073E7	4.775E3	1.042E3	0.970	7.376E7	4.717E3	0.970	2.256E7	7.214E3
		MEAN	6.110E-4	0.986	1.494E8	8.504E3	689.390	0.978	1.514E8	9.027E3	0.976	3.950E7	3.917E3
		BEST	3.906E-4	0.988	2.919E8	1.846E4	373.360	1.000	3.980E8	2.022E4	0.978	6.643E7	2.129E3
CFOA _{AC}	NSGA-II _{DE}	WORST	2.037E-3	0.981	5.425E7	1.297E3	3.113E3	0.971	5.081E7	1.015E4	0.970	1.449E7	8.874E3
		MEAN	5.975E-4	0.987	1.101E8	5.486E3	1.483E3	0.984	1.291E8	6.933E4	0.978	9.835E7	1.861E3
		BEST	2.698E-4	0.988	3.443E8	1.294E4	351.700	1.000	5.167E8	3.378E5	0.984	3.419E8	334.320
	MOEAD _{DE}	WORST	5.503E-3	0.980	6.138E7	386.150	2.226E3	0.970	5.719E7	1.043E3	0.970	2.644E7	4.720E3
		MEAN	3.141E-3	0.983	4.308E8	2.061E4	786.025	0.976	4.173E8	6.598E6	0.977	3.538E8	577.320
		BEST	3.155E-4	0.988	7.002E8	2.154E5	147.580	1.000	6.539E8	1.175E8	0.986	7.155E8	15.396
	MOPSO	WORST	5.866E-3	0.974	1.358E8	8.521E3	803.290	0.970	7.441E7	471.940	0.970	8.286E7	1.168E3
		MEAN	3.108E-3	0.980	3.529E8	1.842E4	329.317	0.982	2.243E8	1.546E3	0.980	3.030E8	375.474
		BEST	1.061E-3	0.984	7.929E8	5.760E4	123.750	1.000	6.324E8	7.065E3	0.987	6.687E8	113.830

Table 3.16: Results of optimization for CFOA_{BA}, CFOA_{BB} and CFOA_{BC}

CFOA	Method		POWER Watts	GAIN _X $ \frac{V_X}{V_Y} $	BW _X Hz	Z _Y Ω	Z _X Ω	GAIN _Z $ \frac{V_Z}{V_X} $	BW _Z Hz	Z _Z Ω	GAIN _W $ \frac{V_X}{V_Y} $	BW _W Hz	Z _W Ω
CFOA _{BA}	NSGA-II _{DE}	WORST	9.102E-3	0.977	2.253E7	6.772E3	6.495E3	0.970	3.319E7	62.267	0.970	5.668E7	3.237E3
		MEAN	2.760E-3	0.985	1.071E8	5.177E4	1.508E3	0.986	1.623E8	1.832E3	0.979	5.118E8	292.173
		BEST	3.354E-4	0.987	3.767E8	3.030E6	295.400	1.000	1.090E9	3.026E4	0.988	1.335E9	34.123
	MOEAD _{DE}	WORST	0.015	0.981	1.577E7	3.342E3	9.727E3	0.970	3.610E7	29.862	0.970	3.177E7	3.881E3
		MEAN	8.462E-3	0.985	1.867E8	4.308E5	2.358E3	0.981	1.789E8	6.604E3	0.974	1.122E9	479.277
		BEST	4.688E-4	0.987	3.377E8	2.568E6	255.830	1.000	5.448E8	5.892E4	0.989	1.991E9	16.510
	MOPSO	WORST	8.973E-3	0.983	7.907E7	6.214E3	1.271E3	0.970	9.073E7	352.250	0.972	2.277E8	287.930
		MEAN	4.375E-3	0.986	1.513E8	1.064E4	649.810	0.989	1.418E8	592.064	0.982	4.932E8	123.039
		BEST	1.639E-3	0.987	2.300E8	2.197E4	429.100	1.000	2.629E8	1.242E3	0.988	1.269E9	46.300
CFOA _{BB}	NSGA-II _{DE}	WORST	6.835E-3	0.973	7.509E7	3.811E4	1.994E3	0.971	1.976E8	1.007E3	0.900	1.031E8	1.290E3
		MEAN	5.769E-3	0.976	1.151E8	1.537E5	1.431E3	0.972	2.121E8	8.558E3	0.923	4.461E8	733.125
		BEST	4.704E-3	0.979	1.550E8	2.693E5	869.200	0.974	2.267E8	1.611E4	0.946	7.890E8	176.550
	MOEAD _{DE}	WORST	8.456E-3	0.981	1.450E7	1.087E4	1.065E4	0.970	3.987E7	539.720	0.970	5.463E6	2.579E4
		MEAN	5.136E-3	0.983	1.044E8	6.354E5	3.360E3	0.982	9.573E7	2.288E4	0.971	2.657E8	2.385E3
		BEST	3.866E-4	0.986	1.956E8	3.171E6	576.840	1.000	1.414E8	4.300E5	0.977	4.313E8	167.980
	MOPSO	WORST	9.539E-3	0.980	1.477E8	2.220E4	864.060	0.987	1.238E8	410.030	0.921	3.074E8	510.400
		MEAN	7.722E-3	0.981	1.619E8	2.523E4	775.178	0.996	1.357E8	627.022	0.944	4.012E8	283.694
		BEST	6.746E-3	0.982	1.718E8	3.062E4	724.340	1.000	1.465E8	1.079E3	0.962	6.463E8	148.020
CFOA _{BC}	NSGA-II _{DE}	WORST	2.558E-3	0.985	1.633E7	1.131E5	9.094E3	0.980	4.084E7	2.470E3	0.977	1.643E8	920.390
		MEAN	1.706E-3	0.985	3.256E7	5.510E5	6.323E3	0.986	5.603E7	6.049E3	0.980	2.740E8	456.133
		BEST	8.760E-4	0.986	6.436E7	1.158E6	2.059E3	0.995	8.465E7	1.302E4	0.983	4.631E8	134.100
	MOEAD _{DE}	WORST	0.023	0.972	2.178E7	8.405E3	6.880E3	0.970	5.088E7	425.590	0.970	3.077E7	4.665E3
		MEAN	0.011	0.980	1.617E8	2.989E5	1.763E3	0.982	1.330E8	1.700E4	0.977	6.830E8	687.377
		BEST	1.427E-3	0.986	2.623E8	1.712E6	438.310	1.000	2.429E8	1.173E5	0.987	1.387E9	27.954
	MOPSO	WORST	0.011	0.978	1.133E8	3.311E4	1.117E3	0.979	1.325E8	850.450	0.971	4.321E8	281.310
		MEAN	8.814E-3	0.981	1.307E8	4.088E4	915.685	0.994	1.536E8	1.373E3	0.976	5.271E8	145.413
		BEST	6.886E-3	0.983	1.507E8	5.603E4	749.280	1.000	1.962E8	2.436E3	0.979	7.599E8	79.365

Table 3.17: Results of optimization for CFOA_{CA}, CFOA_{CB} and CFOA_{CC}

CFOA	Method		POWER Watts	GAIN _X $ \frac{V_X}{V_Y} $	BW _X Hz	Z _Y Ω	Z _X Ω	GAIN _Z $ \frac{V_Z}{V_X} $	BW _Z Hz	Z _Z Ω	GAIN _W $ \frac{V_X}{V_Y} $	BW _W Hz	Z _W Ω
CFOA _{CA}	NSGA-II _{DE}	WORST	9.023E-3	0.970	2.289E7	1.032E4	6.199E3	0.970	3.414E7	249.690	0.970	3.807E5	4.033E5
		MEAN	3.435E-3	0.977	2.672E8	1.897E5	1.050E3	0.984	3.493E8	9.633E3	0.979	4.340E8	4.238E3
		BEST	3.337E-4	0.986	8.131E8	2.476E6	118.420	1.000	9.980E8	3.018E5	0.989	1.237E9	58.420
	MOEAD _{DE}	WORST	0.017	0.970	1.318E7	3.854E3	1.189E4	0.970	3.694E7	79.794	0.970	2.486E6	6.070E4
		MEAN	7.527E-3	0.977	4.590E8	2.487E9	1.945E3	0.983	4.512E8	7.591E4	0.978	7.709E8	4.727E3
		BEST	3.502E-4	0.985	1.222E9	1.388e+010	61.435	1.000	1.027E9	6.866E5	0.989	1.993E9	20.539
	MOPSO	WORST	5.866E-3	0.974	1.358E8	8.521E3	803.290	0.970	7.441E7	471.940	0.970	8.286E7	1.168E3
		MEAN	3.108E-3	0.980	3.529E8	1.842E4	329.317	0.982	2.243E8	1.546E3	0.980	3.030E8	375.474
		BEST	1.061E-3	0.984	7.929E8	5.760E4	123.750	1.000	6.324E8	7.065E3	0.987	6.687E8	113.830
CFOA _{CB}	NSGA-II _{DE}	WORST	0.012	0.970	3.624E7	8.592E3	4.255E3	0.970	3.502E7	723.430	0.970	4.866E6	3.293E4
		MEAN	4.482E-3	0.979	2.760E8	1.075E5	811.389	0.985	3.026E8	5.342E3	0.975	1.336E8	1.918E3
		BEST	6.926E-4	0.985	1.111E9	1.567E6	90.174	1.000	1.443E9	1.666E5	0.978	3.691E8	332.220
	MOEAD _{DE}	WORST	0.019	0.970	1.658E7	3.532E3	9.402E3	0.970	4.390E7	367.990	0.970	1.455E6	9.879E4
		MEAN	0.012	0.977	5.648E8	1.508E6	2.374E3	0.982	5.552E8	5.958E4	0.974	3.567E8	8.919E3
		BEST	4.305E-4	0.986	1.519E9	1.006E7	41.254	1.000	1.541E9	5.172E5	0.979	9.040E8	94.878
	MOPSO	WORST	8.538E-3	0.977	1.534E8	9.316E3	770.800	0.970	8.270E7	857.640	0.970	7.036E7	1.515E3
		MEAN	4.078E-3	0.982	3.492E8	1.963E4	303.426	0.983	2.075E8	1.504E3	0.976	1.453E8	746.285
		BEST	1.502E-3	0.985	8.174E8	5.689E4	112.950	1.000	7.021E8	3.102E3	0.978	2.549E8	399.840
CFOA _{CC}	NSGA-II _{DE}	WORST	0.016	0.970	3.591E7	6.491E3	3.952E3	0.970	4.663E7	3.252E3	0.970	1.822E7	7.409E3
		MEAN	3.827E-3	0.980	2.519E8	7.932E4	806.524	0.986	2.686E8	2.851E4	0.979	2.612E8	1.040E3
		BEST	7.449E-4	0.986	1.117E9	2.009E6	87.973	1.000	1.366E9	1.763E5	0.985	1.104E9	74.421
	MOEAD _{DE}	WORST	0.037	0.970	2.219E7	2.249E3	6.972E3	0.970	5.317E7	1.020E3	0.970	3.722E7	5.151E3
		MEAN	0.017	0.978	6.030E8	5.888E7	915.536	0.982	5.567E8	1.579E5	0.978	8.355E8	477.254
		BEST	7.264E-4	0.986	1.552E9	8.659E8	34.850	1.000	1.539E9	3.876E6	0.985	2.042E9	15.463
	MOPSO	WORST	9.544E-3	0.980	1.347E8	6.292E3	867.010	0.970	6.380E7	4.062E3	0.972	9.268E7	1.616E3
		MEAN	4.834E-3	0.983	2.798E8	1.235E4	409.508	0.987	1.337E8	7.739E3	0.982	2.489E8	418.152
		BEST	1.568E-3	0.985	5.615E8	2.896E4	209.210	1.000	3.435E8	3.030E4	0.984	5.138E8	166.430

3.5 Summary

This chapter has shown the proposed methodology to optimize analog circuits by using EAs. Also it is described how the simulator measures the circuit performances to evaluate the different objective functions. There was performed the optimization of two mixed mode circuits, with different number of variables and three objectives for voltage and three objectives for current, along several runs to ensure the repeatability of the results. However all the EAs with both recombination operator (SBX and DE) have optimal results, no method dominates completely to another method.

The last experiments were performed to optimize nine different CFOAs with different number of variables each one and for eleven objective functions in voltage and current mode. As before, there is no dominance percentage of any method over another one, however in general, the best values was found by MOEAD that improve slightly the NSGA-II and the MOPSO values, but always there exists some objectives where MOEAD can not improve over NSGA-II and MOPSO.

Finally, we can conclude that the proposed methodology is able to find optimal solutions nevertheless there is not one EA nor recombination operator that dominates completely to the others.

Chapter 4

Automatic current-bias distribution

4.1 Introduction

It is presented an analog circuit relative auto-biasing methodology, based on a recursive technique for exploring graphs. The concept of nullor is used to biasing the MOSFETs along the different branches of a circuit. The aim is to bound the search space of each transistor according with the portion of current needed to preserve its biasing in a optimization process. There are presented examples that show the usefulness of this methodology in the multi-objective optimization of analog circuits.

4.2 Bias assignments in CMOS analog circuits by graph manipulations

The analog circuits optimization research has grown a lot in the last ten years, by exploiting not only deterministic techniques, but also heuristic ones. In all of them, it is necessary to specify: the design variables, the search space, the constraints and the objectives, among other parameters.

If we focus on the specification of the design variables, that process requires to have a previous insight on the circuit being optimized, because some variables such as the width or length of transistors can be (some times needs to be) the same for some of them, this process

is called encoding. The encoding of a circuit, allows grouping several design variables into only one variable in the optimization process. The encoding can be the key of a successful and short-time optimization, an unsuitable encoding instead, might lead us to a slow optimization or indeed to a failed optimization [84].

The circuit biasing is the process through which the voltage and currents of the elements of an electronic circuit are set, with the aim to ensure its proper operation. This process depends mainly of the supply voltage and the reference bias current [85].

The circuit sizing can be defined as a process through which the the dimensions of the width and length of the transistors are set. There exist different ways to make the circuit sizing, being the most basic hand calculation, taking into account the technologies parameters and the current equation of the transistor. But also there exist a lot of different options regarding circuit biasing and sizing such as deterministic and heuristic techniques. The sizing has a direct impact on the biasing, because the dimensions of the transistor determine the voltage on its source and drain terminals trying to preserve the current of the branch where the transistor is.

Sometimes it is necessary to have a deep insight in a circuit to achieve a good sizing, which in turn properly bias it , and this can take a lot of time. Part of this insight, has relation with the search space for each encoded variable within an optimization process, because setting properly the limits over the search space for each variable allows the optimizer to find an optimal solution, without wander over the entire space and wasting a lot of time in such task with the risk to fail the optimization, if the number of variables is large.

Using an autobiasing methodology in automated sizing can have several benefits: faster algorithm convergence, better-quality results, better designer insight, improved acceptance of automated sizers by designers, and more. Different autobiasing methodologies are used in ADA tools such in:

- Hierarchical Automated Sizing [8, 86, 87].
- Classification / Regression [88, 89].
- Symbolic Modeling [24, 90].
- Topology Selection / Synthesis [91].
- Variation-Aware Automated Sizing [27].

4.2. BIAS ASSIGNMENTS IN CMOS ANALOG CIRCUITS BY GRAPH MANIPULATIONS 61

- Portable Models of Circuit Performance [92].
- New Manual Sizing Flows [93, 94].

Despite the enormous benefit that provides an autobiasing methodology, yet there exists some issues that hinder its implementation into a sizing process. Usually, all the autobiasing approaches have a specific application, have a high complexity, are transistor models dependent and they handle a specific design variables space.

Then, we propose an analog circuit relative auto-biasing methodology, based on a recursive technique for exploring graphs associating bias-currents for each transistor. The concept of nullor is used to distribute the current-bias of the MOSFETs along the different branches of a circuit. The aim is to bound the search space of each transistor according to the portion of current needed to preserve its biasing in a optimization process.

The basic idea is to choose the appropriate limits over the search space for the encoded variables in the sizing process that takes place in the analog circuit optimization. In this manner it can be possible to achieve a successful optimization without waste of time on searching values which are not feasible due to the wrong biasing of the circuit, and without having a deep insight in the circuit.

The proposed method has the advantage that can be used within any optimization process (deterministic or heuristic) that could handle different search spaces for the different variables. This method does not interfere with the sizing process, because it takes place only one time before the optimization begins, i.e. it is like an starting point for establishing the bound limits for the variables to be sized. For the specific case of circuit optimization, the auto-biasing current partitioning takes less or similar time than a circuit simulation, then the cost of this method is minimum compared with the large number of simulations that the optimization process will save only by setting the limits in the search space for each variable.

4.3 Modeling the Transistor for Current Bias Partitioning

4.3.1 Modeling the Transistor

The biasing and sizing problem in this work will be oriented mainly to the transistors which make up a circuit. Then, to manage easily the transistors we apply a model proposed in [95], where the concept of nullor is useful to reduce the transistor to a two ports element. In Figure 4.1(b) is depicted the nullator, a theoretical element that exhibits zero voltage and zero current. In Figure 4.1(b) is depicted the norator, another theoretical element that exhibits an arbitrary voltage and current. The nullor is the element composed by a norator and a nullator as depicts Figure 4.1(c).

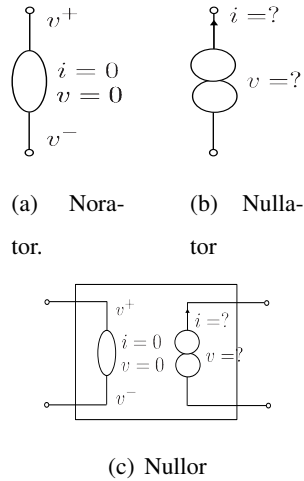


Figure 4.1: Norator, nullator and nullor

By using the concept of nullor it is possible to represent the N-MOSFET as depicts Figure 4.2. Taking into account that nullator exhibits zero voltage and zero current, the GATE port can be neglected due that any current does not enter or outgoing it. This feature is agreed with the high resistance in the GATE port of a MOSFET and allows modeling the current behavior of the transistor as a two ports (DRAIN and SOURCE) element as Figure 4.3 shows.

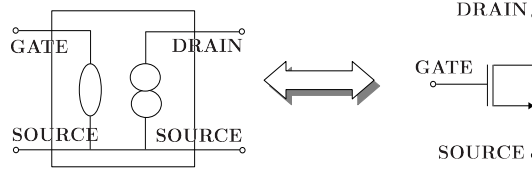


Figure 4.2: Modeling a MOSFET with a nullor.

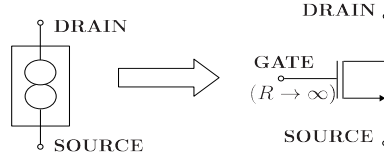


Figure 4.3: Current behavior model of a MOSFET.

4.3.2 Current Bias Distribution

The Kirchhoff's current law indicates that the algebraic sum of the branches currents entering or outgoing from a node is zero [96]. In the case that a node has only one entering current branch, and only one current branch outgoing from it, the current in both branches are the same (Figure 4.4(a)). When there are more than one outgoing branches from a node, the current is not always symmetrically distributed over the branches that leave of such node, however it is accomplished the current law as shows Figure 4.4(b), where the sum of the currents leaving the node is equal to the currents arriving to the node ($i_a = i_b + i_c + i_d$). Then, it is possible to consider that currents i_b , i_c and i_d are a portion of i_a , in this manner it is possible to rewrite i_b as $\alpha \cdot i_a$, i_c as $\gamma \cdot i_a$ and i_d as $\delta \cdot i_a$, where α , γ and δ are real positive numbers less than one and their sum is equal to one.

There may exist different current distributions over the outgoing branches in a node, because different performances of the circuit demands different distributions on the current branches. But according to the current law, it is possible to know that the outgoing branches of a node has a portion of the entering currents.

This work proposes a method which is based on the distribution of the current bias over all the trajectories from V_{dd} to V_{ss} . The aim is to bound the search space of each element according to the portion of current needed to preserve the biasing in a optimization process. In the next section it will be exposed the way to traverse the circuit to find the different current branches

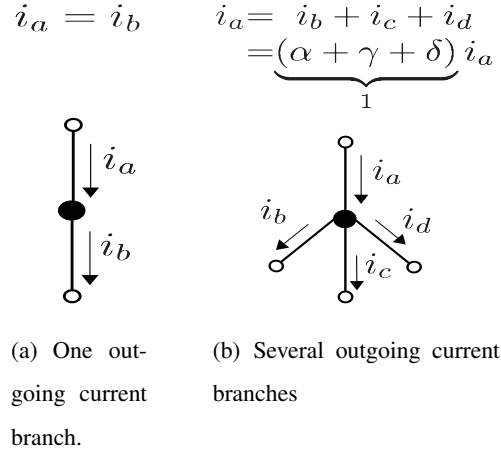


Figure 4.4: Examples of outgoing currents branches.

and how the current bias is distributed in each element.

4.4 Circuits and graphs

4.4.1 Incidence Matrix

From a given circuit it can be build a directed graph $G = \langle \mathbf{N}, \mathbf{B} \rangle$ where \mathbf{N} is the set of nodes and \mathbf{B} the set of current branches. In order to build that graph, it is used the “Incidence Matrix” [97] denoted by $\mathbf{A}_{k \times l}$ where its rows represent the nodes $\mathbf{N} = \{n_1, n_2, \dots, n_k\}$ (all nodes except the reference node as V_{ss} or ground) and its columns represent the branches (circuit elements) $B = \{b_1, b_2, \dots, b_l\}$. The values of each element a_{ij} in \mathbf{A} are 0, 1 or -1 according to (4.1).

With the matrix A it is possible to build a directed graph that represents a circuit, in the following way:

- The graph has k nodes and l branches.
- A 1 in the i^{th} row and the j^{th} column means that the branch j leaves the node i .
- A -1 in the i^{th} row and the j^{th} column means that the branch j enters to the node i .
- A 0 (or empty) in the i^{th} row and the j^{th} column means that the branch j does not enter or leave the node i .

$$\mathbf{A} = \begin{matrix} & \begin{matrix} b_1 & b_2 & \dots & b_l \end{matrix} \\ \begin{matrix} n_2 \\ n_2 \\ \vdots \\ n_k \end{matrix} & \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1l} \\ a_{21} & a_{22} & \dots & a_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \dots & a_{kl} \end{bmatrix} \end{matrix} \quad (4.1)$$

$$\text{where } a_{ij} = \begin{cases} 1 & \text{if } b_j \text{ leaves } n_i \\ -1 & \text{if } b_j \text{ arrives to } n_i \\ 0 \text{ or empty} & \text{otherwise} \end{cases}$$

4.4.2 Depth First Search for biasing

Once the graph is described with the matrix \mathbf{A} and the above issues, it is possible to explore the graph to find all the current trajectories. Then it is used a “Depth First Search” (*dfs* algorithm) [98] shown in Algorithm 9. There is a vector labeled as *Visited_flag* which associates a flag to each branch in order to know whether it has been already visited and avoid visiting it twice. This recursive algorithm explores all the adjacent branches to a given branch. It is considered the adjacent branches of a branch b , those branches which share the same node as b .

Algorithm 9 Depth First Search Algorithm (*dfs*)

Input: a branch b

- 1: *visited_flag*[b] \leftarrow visited
 - 2: **for each** branch b_i adjacent to b **do**
 - 3: *dfs*(b_i) **if** *Visited_flag*[b_i] \neq visited
 - 4: **end for**
-

The *dfs* algorithm is modified to find the different distributions (*Levels*) of the current bias in each branch represented by an element (transistor) of the circuit. Then we proposed a top-down algorithm *dfs_{TD}* shown in Algorithm 10 which has as arguments: the current branch b , the upper node of b namely n and the bias level *CurLevel*. As before, there exists a vector flag called as *Visited* which associates a flag to each branch, but now, there is other more vector

Algorithm 10 Depth First Search Algorithm Top-Down (dfs_{TD})

Input: $b, n, CurLevel$

```

1:  $Visited[b] \leftarrow \text{visited}$ 
2:  $\mathbf{B}_n \leftarrow$  set of outgoing branches from  $n$  different of  $b$ 
3: if  $\mathbf{B}_n \neq \emptyset$  then
4:   for each branch  $b_i \in \mathbf{B}_n$  do  $CurLevel- = 1$ 
5: else
6:    $CurLevel \leftarrow$  level of the upper branch of  $b$ 
7: end if
8:  $Bias\_Level[b] \leftarrow CurLevel$ 
9:  $n \leftarrow$  lower node of  $b$ 
10: for each entering branch to  $n$  do  $CurLevel+ = 1$ 
11:  $\mathbf{B}_n \leftarrow$  set of outgoing branches from  $n$ 
12: for each branch  $b_i \in \mathbf{B}_n$  do
13:    $dfs_{TD}(b_i, n, CurLevel)$  if  $Visited[b_i] \neq \text{visited}$ 
14: end for
```

called $Bias_level$ which associates the bias level to each branch. This algorithm traverses the circuit in a top-down approach: from V_{dd} to V_{ss} .

The first step in Algorithm 10 is to mark the branch b as visited one. The second line stores in \mathbf{B}_n the outgoing branches from n different of b . Next, line 3 evaluates whether \mathbf{B}_n is different from empty to subtract one to $CurLevel$ for each branch b_i in \mathbf{B}_n . If \mathbf{B}_n is an empty set, it means that there are not adjacent branches to b , and therefore, the branch b has the same level than its upper branch. In the line 8, the level ($CurLevel$) is assigned to the branch b . Line 9 sets to n the lower node of b . Afterwards, the line 10 finds the branches that enter to the new node n and for each one of them, it is added one to $CurLevel$. In line 11, it is repeated the procedure of line 2 (with the new node n), and finally in line 12, there is a recurrence of the dfs_{TD} to itself, if b_i has not been visited. In this manner, it is possible to find the current level in a given branch.

Process described in lines 2, 6, 9, 10 and 11, are performed by using the incidence matrix

(**A**), because that matrix contains all the information about the nodes, branches, their connections and directions.

4.5 Proposed Current-Banches-Bias Assignment (CBBA) Approach

Our proposed Current-Banches-Bias Assignment (CBBA) approach distributes the current bias reference(s) according to the depth of each element (branch), and it can be summarized as follows:

1. Replace all transistors by their nullor equivalent (see Figure 4.3).
2. Build the incidence matrix **A** of the equivalent circuit, i.e. (4.1).
3. Build ($G = \langle \mathbf{N}, \mathbf{B} \rangle$) from the incidence matrix **A**.
4. Apply recursively Algorithm 10 .

Algorithm 11 describes the above steps: from a SPICE circuit netlist, in line 1 matrix **A** is generated as in (4.1). Next, from line 2 to line 5, the vector flag *Visited* is initialized to control the recursive calls and the flag vector *Bias_level*. The distributed or partitioned level is stored as a result of the auto-biasing process. In line 6 the outgoing branches from V_{dd} , are stored in $\mathbf{B}_{V_{dd}}$. Next, for each branch b_i in $\mathbf{B}_{V_{dd}}$ the *CurLevel* is initialized with zero, the method sets the zero level to b_i and labels it as a visited branch. In line 11, the lower node of b_i is stored in n , with the aim to build vector \mathbf{B}_n formed by all the outgoing branches from node n in line 12. In lines 13 to 15 there is a recursive call to dfs_{TD} for each branch b_j , if it has not been already visited. At last, each branch has been assigned a level, where the lower level requires more part of the space search (in other words, it holds more current than in the rest of levels).

When each branch has a current bias level, it is possible to perform a procedure to set the limits of the search space for each variable. We propose an heuristic procedure to assign a quantity proportional to the current bias level: let X_l be the lower and X_u the upper limits of the whole search space, L_l^k the low limit and L_u^k the upper limits in the search space for the k -th level.

Algorithm 11 Distribution of Currents**Input:** A circuit netlist, specifying V_{dd} and V_{ss} node

-
- 1: Build the matrix \mathbf{A} and the graph $G = \langle \mathbf{N}, \mathbf{B} \rangle$
 - 2: **for each** branch $b_i \in \mathbf{B}$ **do**
 - 3: $Visited[b_i] \leftarrow$ not visited
 - 4: $Bias_level[b_i] \leftarrow 0$
 - 5: **end for**
 - 6: $\mathbf{B}_{V_{dd}} \leftarrow$ set of branches outgoing from node V_{dd}
 - 7: **for each** branch $b_i \in \mathbf{B}_{V_{dd}}$ **do**
 - 8: $CurLevel \leftarrow 0$
 - 9: $Bias_level[b_i] \leftarrow 0$
 - 10: $Visited[b_i] \leftarrow$ visited
 - 11: $n \leftarrow$ the lower node of b_i
 - 12: $\mathbf{B}_n \leftarrow$ set of outgoing branches from n
 - 13: **for each** branch $b_j \in \mathbf{B}_n$ **do**
 - 14: $dfs_{TD}(b_j, n, CurLevel)$ **if** $Visited[b_j] \neq$ visited
 - 15: $CurLevel = 0$
 - 16: **end for**
 - 17: **end for**
-

Algorithm 12 describes the assignment procedure. It consists of dividing the search space into sub-spaces according to the total number of levels (TL). The first step is devoted to divide the entire search space into two parts, corresponding to the two first levels (level 0 and level 1). Those two parts share an intersection region to relax the partitioning and allow exploring beyond the bound limits. The intersection between two levels (ν_\cap) depends on the total number of levels and can be controlled by using an integer scaling factor (χ) as shown by (4.2). Since $TL > 0$ and $\chi > 0$, then $\nu_\cap < 0.5$. In our experiments, we used $\chi = 1$.

$$\nu_\cap = (TL + \chi)^{-1} \quad (4.2)$$

Algorithm 12 Limit search space assignment procedure

Input: X_l, X_u, TL

```

1:  $k = 0$ 
2:  $X_{u'} = X_u$ 
3:  $\nu_\cap = (TL + \chi)^{-1}$ 
4: while  $k < TL$  do
5:    $X_l^k = X_{u'} * \nu_\cap$ 
6:    $X_u^k = X_{u'}$ 
7:    $X_l^{k+1} = X_l$ 
8:    $X_{u'} = L_{aux} * (1 - \nu_\cap)$ 
9:    $X_u^{k+1} = X_{u'}$ 
10:   $k+ = 2$ 
11: end while
12: if  $TL$  is odd then
13:    $X_l^{TL} = X_l$ 
14:    $X_u^{TL} = X_{u'}$ 
15: end if

```

In this manner the first loop in Algorithm 12 (lines 5 to 10) generates a result like Figure 4.5(a). For the second loop the process is repeated but this time the upper limit is bounded by $X_{u'}$ as shown in Figure 4.5(b). The algorithm continues until splitting the search space for all the levels.

When the number of levels is odd, Algorithm 12 assigns to the last level X_l , its lower limit, and the last value of $X_{u'}$ to its upper limit. In Figure 4.6 is depicted the result of Algorithm 12 for 5 levels ($TL = 5$).

All these CBBA's and bound limits in the search space can be used within an optimization process by choosing different values for the sizing of each MOSFET. The main advantage is preserving the DC operating point of the circuit.

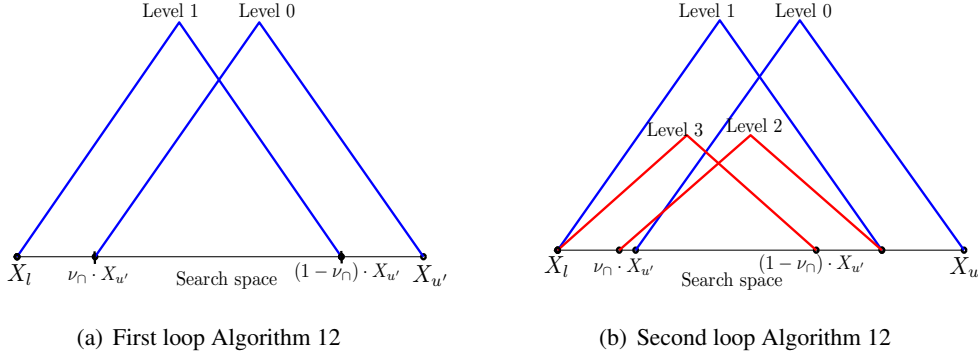


Figure 4.5: First two loops of Algorithm 12.

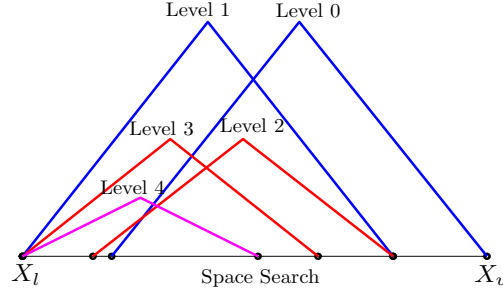


Figure 4.6: Limit search space assignment.

4.6 Application Examples

Our proposed CBBA approach is tested in optimizing two analog circuits: the Folded Cascode (FC) Operational Transconductance Amplifier (OTA) shown in Figure 4.7, and the Recycled Folded Cascode (RFC) OTA shown in Figure 4.8.

The FC OTA is encoded with seven variables (design parameters) for the MOSFETs, $\mathbf{x} = [x_1, \dots, x_7]^T$ as shown in the left side of Table 4.1, and the RFC OTA with ten variables, $\mathbf{x} = [x_1, \dots, x_{10}]^T$ as shown in the right side of Table 4.1. Both circuits are biased with $I_{ref} = 400\mu A$ and $V_{DD} = 1.8V$. The measurements were executed with a load capacitor of 5.6pF and HSPICE[®] simulations were performed with a LEVEL 54 standard CMOS Technology of 90 nm.

For the FC and RFC OTAs is used the circuit depicted on Figure 4.9 for yielding the voltages: Vbn1, Vbn2, Vbp1 and Vbp2. In Table 4.2 are listed the sizes for each transistor of the

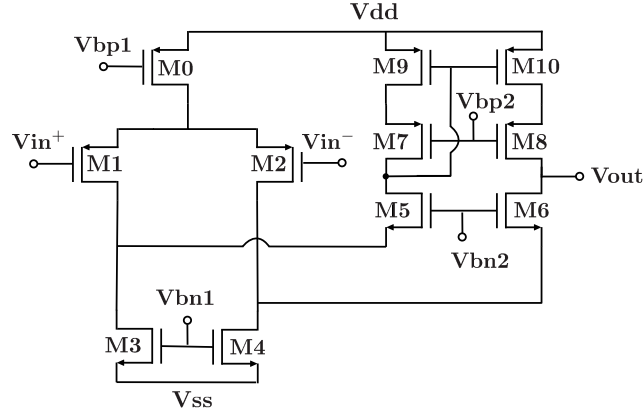


Figure 4.7: Folded Cascode OTA.

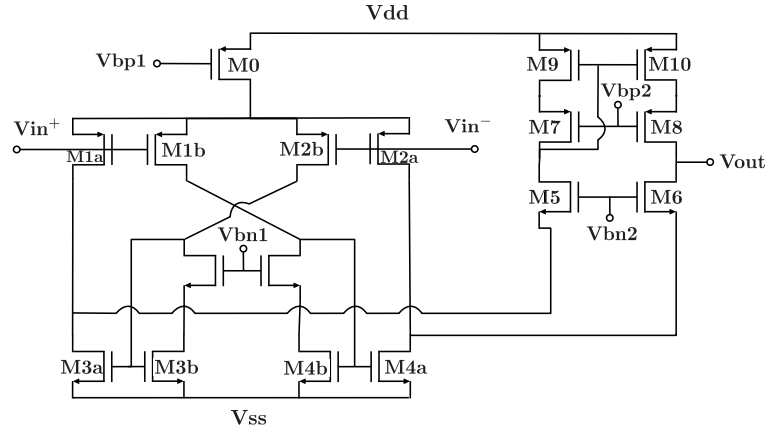


Figure 4.8: Recycled Folded Cascode OTA taken from [99].

voltage references as a dependency of the design variables $L1$, $W1$ and $W5$. The voltage references circuit is not included into the CBBA analysis due to that this circuit can be replaced by any other (indeed by simple voltage sources) which set the bias voltages required for the OTAs.

To show the usefulness of the proposed CBBA approach, the experiments were performed with three different evolutionary algorithms: NSGA-II [76], MOEAD [66] and MOPSO [79], and by using two genetic operators: simulated binary crossover (SBX [70]) and differential evolution (DE [73]).

Table 4.1: Encoding for the FC OTA and RFC OTA.

<i>gene</i>	Design Variable	Encoding Transistors	
		FC OTA	RFC OTA
x_1	L_1	M0,M3,M4,M9,M10	M0,M3a,M3b,M4a,M4b,M9,M10
x_2	L_2	M5, ..., M8	M5, ..., M8
	$2L_2$	M1,M2	M1a,M1b,M2a,M2b
x_3	W_1	M0, M1, M2	M0
x_4	W_2	M3, M4	M1a,M1b,M2a,M2b
x_5	W_3	M5, M6	M3a,M4a
x_6	W_4	M7, M8	M3b,M4b
x_7	W_5	M9, M10	M5, M6
x_8	W_6	-	M7, M8
x_9	W_7	-	M9, M10
x_{10}	W_8	-	M11, M12

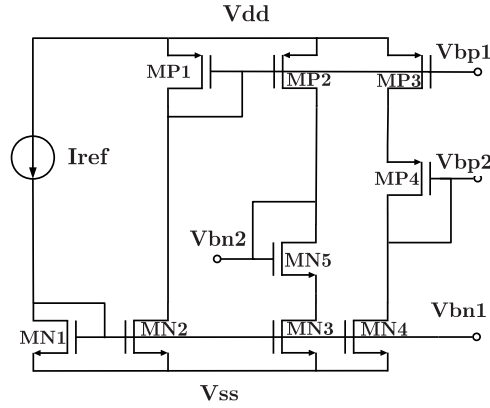


Figure 4.9: Voltage references for Vbn1, Vbn2, Vbp1 and Vbp2.

4.6.1 FC OTA Results

The sizing of the FC OTA is performed to optimize five target specifications: DC gain, gain bandwidth product (GBW), input offset, settling time (ST), and power consumption (PW). Equation (4.3) shows the multi-objective optimization problem as the vector formed by the

Table 4.2: Transistor sizes for voltage references.

Transistor	L	W
MN1	L_1	$3 \cdot W_5$
MN2	L_1	$3 \cdot W_5$
MN3	L_1	$1.5 \cdot W_5$
MN4	L_1	$1.5 \cdot W_5$
MP1	L_1	$1.5 \cdot W_1$
MP2	L_1	$6 \cdot W_5$
MP3	L_1	$6 \cdot W_5$
MP4	L_1	$3 \cdot W_5$

five objectives.

$$\begin{aligned}
 & \text{minimize } \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_5(\mathbf{x})]^T \\
 & \text{subject to } h_l(\mathbf{x}) \geq 0, \quad l = 1 \dots p, \\
 & \text{where } \mathbf{x} \in X.
 \end{aligned} \tag{4.3}$$

In (4.3), $X : \mathbb{R}^n \mid 0.18 \mu m \leq L_i \leq 0.9 \mu m, 0.9 \mu m \leq W_j \leq 130 \mu m$, is the decision space for the n variables, and $h_l(\mathbf{x})$, $l = 1 \dots p$ are performance constraints. In our experiments we included the saturation condition in all transistors and the five target specifications as constraints. $\mathbf{f}(\mathbf{x})$ is the vector formed by:

- $f_1(\mathbf{x}) = -1 \cdot \text{Gain}$.
- $f_2(\mathbf{x}) = -1 \cdot \text{GBW}$.
- $f_3(\mathbf{x}) = \text{Voltage offset}$.
- $f_4(\mathbf{x}) = \text{Settling time}$.
- $f_5(\mathbf{x}) = \text{Power consumption}$.

The optimization procedure is performed with a population size of 210 along 250 generations. After performing the CBBA approach, the current levels are listed in Table 4.3. They are: level 0 assigned to M0, M3, M4 and level 1 to M6, M7, M9, M2, M8, M1, M10 and M5. According to Table 4.3, M1 and M2 have Level 1 (because the current from M0 is divided through

M1 and M2). However, according to the encoding (left side of Table 4.1), those transistors have the same design variable as M0 (Level 0). In this case, the system identifies the same design variable ($W1$) for M0, M1 and M2, and takes the lower current level among those transistors.

Next, Algorithm 12 is executed to determine the current limits, as shown in Figure 4.10. Finally, Tables 4.4 to 4.6 show the optimized results for the FC OTA for NSGA-II, MOEAD and MOPSO with and without CBBA. For those results, there were selected among the non-dominated solutions those which accomplish with the less power consumption.

Table 4.3: Current-branches-bias assignments for the FC OTA.

Level	Elements	Upper Bound [μm]	Lower Bound [μm]
0	M3, M4, M0	130	43
1	M6, M7, M9, M2, M8, M1, M10, M5	86	0.9

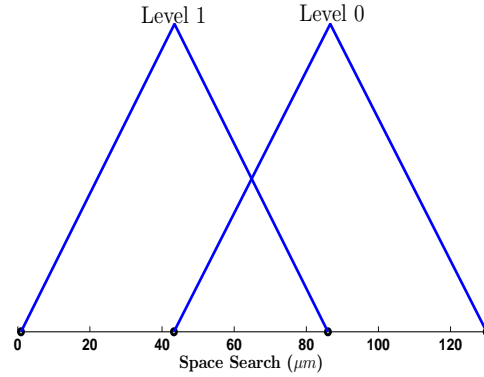


Figure 4.10: Search Space Limits for the FC OTA.

On the left side of Table 4.4 are depicted the optimized results provided by NSGA-II with SBX as genetic operator (NSGA_{SBX}). In this case, the performances are improved using CBBA, except for the offset (stays in the same value) and power consumption. However all of them accomplish target specifications. The bottom of Table 4.4 lists the generations at which the optimal solution is found. As one sees, the application of CBBA helps to find the optimal solution faster. Figure 4.11(a) depicts the behavior of the solutions along the generations, and

when CBBA is used, the number of biased solutions increase.

The optimal solutions for NSGA_{DE} are depicted on the right side of Table 4.4. The optimal solutions without CBBA do not accomplish the power consumption specification, but the use of CBBA overcomes that issue at generation 221. Regarding to the number of solutions, Figure 4.11(b) depicts a slight increase, showing that CBBA allows accomplishing all target specifications.

The results provided by MOEAD_{SBX} are listed on the left side of Table 4.5. Without CBBA, the optimal solution exhibits a fail around 15% in the GBW value compared with the target specification (70 MHz). When CBBA is used, the optimal solution accomplishes the GBW and all the circuit performances are improved except for the power consumption that shows an increment of 5% compared with the target specification. The number of biased solutions for this experiment increased 4x when CBBA is used, as shown in Figure 4.12(a).

MOEAD_{DE} results (right side of Table 4.5) exhibits a similar behavior regarding to the objective values, but this time, the GBW fails almost 50% without CBBA. Using CBBA achieves accomplishing the GBW specification, although the gain is 2% below the specification and the power consumption is 10% higher than specification. Therefore, CBBA helps to find more solutions and more close to the specifications, as shown in Figure 4.12(b).

Table 4.6 shows the objective values for the optimal solutions with MOPSO. This time, CBBA preserves the circuit performances (except for settling time and power that exhibit a slight increment), but the optimal solution is found around 30 generations faster. Figure 4.13 shows that all the population achieves solutions almost 70 generations faster when CBBA is used.

Table 4.4: Optimal solutions for the FC OTA with NSGA-II.

	Specs.	NSGA _{SBX}		NSGA _{DE}	
		Without	With	Without	With
		CBBA	CBBA	CBBA	CBBA
Objectives					
Gain [dB]	≥ 46	52	53	49	50
GBW [MHz]	≥ 70	71	73	72	70
Offset [<i>mV</i>]	≤ 7.9	5.1	5.1	6.4	5.8
ST [ns]	≤ 20	16.4	15.8	14.8	16.2
PW [mW]	≤ 3.9	3.6	3.9	4.3	3.9
Variables					
<i>L1</i> [<i>μm</i>]	0.18	0.18	0.18	0.22	0.20
<i>L2</i> [<i>μm</i>]	0.5	0.20	0.23	0.18	0.23
<i>W1</i> [<i>μm</i>]	128	129.28	129.90	127.08	126.10
<i>W2</i> [<i>μm</i>]	32	32.20	43.76	52.84	43.29
<i>W3</i> [<i>μm</i>]	16	15.03	15.28	14.03	14.30
<i>W4</i> [<i>μm</i>]	64	121.46	85.48	76.31	36.10
<i>W5</i> [<i>μm</i>]	64	6.19	35.59	29.44	78.94
Generation		238	167	55	221

Table 4.5: Optimal solutions for the FC OTA with MOEAD.

	Specs.	MOEAD _{SBX}		MOEAD _{DE}	
		Without CBBA	With CBBA	Without CBBA	With CBBA
Objectives					
Gain [dB]	≥ 46	48	49	58	45
GBW [MHz]	≥ 70	60	72	37	72
Offset [mV]	≤ 7.9	7.3	4.8	6.7	4.3
ST [ns]	≤ 20	17.17	16.29	29.8	16.3
PW [mW]	≤ 3.9	3.6	4.1	3.0	4.4
Variables					
$L1[\mu m]$	0.18	0.26	0.18	0.31	0.18
$L2[\mu m]$	0.5	0.21	0.20	0.62	0.22
$W1[\mu m]$	128	100.02	120.55	94.44	127.59
$W2[\mu m]$	32	28.32	43.33	25.42	53.80
$W3[\mu m]$	16	11.19	13.12	12.24	12.90
$W4[\mu m]$	64	104.49	51.62	46.90	61.80
$W5[\mu m]$	64	24.44	40.23	2.95	85.44
Generation		146	135	55	139

Table 4.6: Optimal solutions for the FC OTA with MOPSO.

	Specs.	MOPSO	
		Without CBBA	With CBBA
Objectives			
Gain [dB]	≥ 46	52	52
GBW [MHz]	≥ 70	74	74
Offset [mV]	≤ 7.9	5.2	6.0
ST [ns]	≤ 20	15.9	14.5
PW [mW]	≤ 3.9	3.6	3.9
Variables			
$L1[\mu m]$	0.18	0.19	0.20
$L2[\mu m]$	0.5	0.18	0.18
$W1[\mu m]$	128	129.92	129.90
$W2[\mu m]$	32	32.11	43.33
$W3[\mu m]$	16	15.75	14.78
$W4[\mu m]$	64	16.09	79.86
$W5[\mu m]$	64	10.76	36.13
Generation		249	217

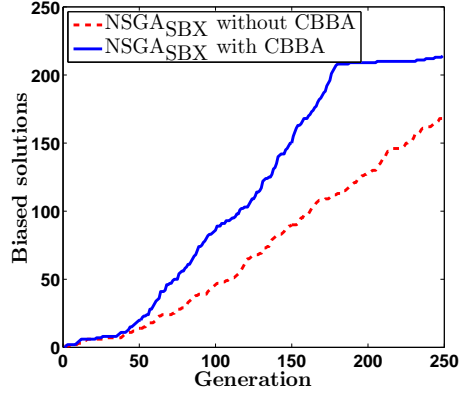
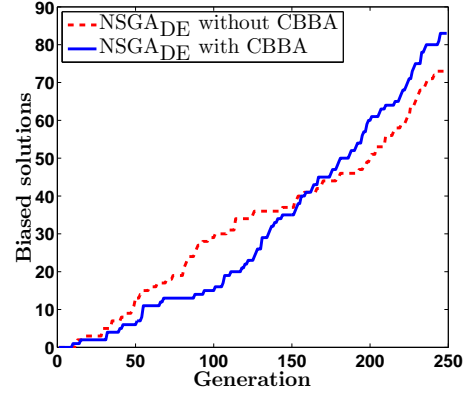
(a) NSGA-II_{SBX} Optimization.(b) NSGA-II_{DE} Optimization.

Figure 4.11: Solutions for the FC OTA with and without CBBA for NSGA-II.

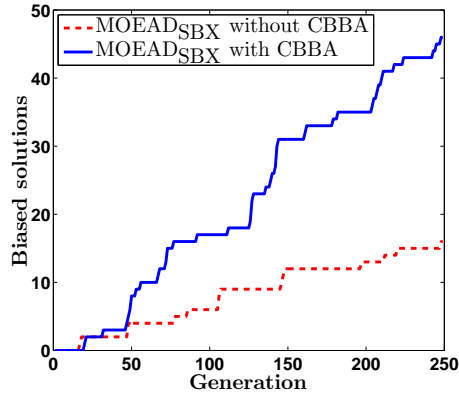
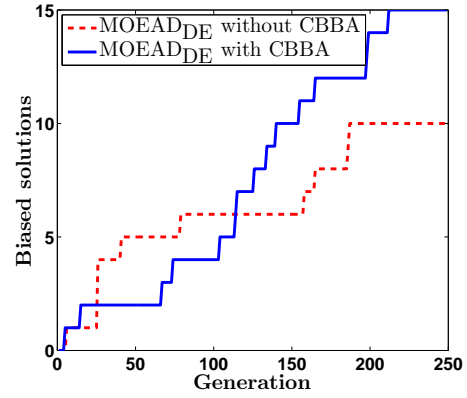
(a) MOEAD_{SBX} Optimization.(b) MOEAD_{DE} Optimization.

Figure 4.12: Solutions for the FC OTA with and without CBBA for MOEAD.

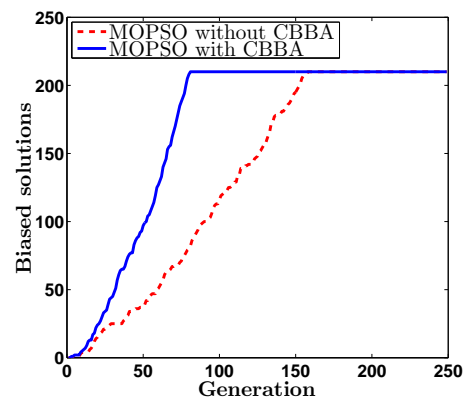


Figure 4.13: Solutions for the FC OTA with and without CBBA for MOPSO.

4.6.2 RFC OTA Results

The sizing of the RFC OTA is performed to optimize eight target specifications: DC gain, GBW, PM, input referred noise, input voltage offset, ST, SR and PW. Equation (4.3) is also used for this multi-objective optimization problem. But now $\mathbf{f}(\mathbf{x})$ is the vector formed by eight objectives:

- $f_1(\mathbf{x}) = -1 * \text{Gain}$.
- $f_2(\mathbf{x}) = -1 * \text{GBW}$.
- $f_3(\mathbf{x}) = -1 * \text{PM}$
- $f_4(\mathbf{x}) = \text{Input referred noise}$.
- $f_5(\mathbf{x}) = \text{Input voltage offset}$.
- $f_6(\mathbf{x}) = \text{Settling time}$.
- $f_7(\mathbf{x}) = -1 * \text{Slew rate}$.
- $f_8(\mathbf{x}) = \text{Power consumption}$.

In this experiment, also $X : \mathbb{R}^n \mid 0.18 \mu m \leq L_i \leq 0.9 \mu m, 0.9 \mu m \leq W_j \leq 130 \mu m$, is the decision space for the n variables, and $h_l(\mathbf{x})$, $l = 1 \dots p$ are performance constraints. Again, we include the saturation condition in all transistors and the target specifications for the eight objectives as constraints.

Table 4.7: Current-branches-bias assignments to the RFC OTA.

Level	Elements	Upper Bound [μm]	Lower Bound [μm]
0	M0	130	26
1		104	0.9
2	M3A, M4A	104	20
3	M6, M11, M1B, M7, M9, M12, M2B, M8, M1A, M2A, M10, M5, M3B, M4B	83	0.9

The optimization procedure is performed with a population size of 210 along 250 generations. After executing the proposed CBBA approach, the current levels are listed in Table 4.7. The approach assigned four bias levels: level 0 is assigned to M0, level 1 does not have elements, level 2 is conformed by M3A, M4A and Level 3 has the rest of transistors.

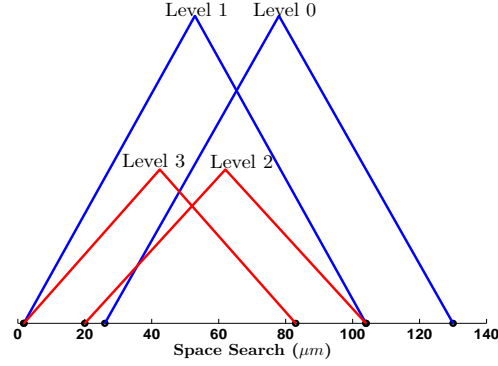


Figure 4.14: Search Space Limits for the RFC OTA.

Next, by applying Algorithm 12, the limits in the search space for each current level are set as shown in Figure 4.14. Tables 4.8 to 4.10 show the results for the RFC OTA provided by NSGA-II, MOEAD and MOPSO, with and without CBBA, and by using SBX and DE as genetic operators. As before, for those results, there were selected among the non-dominated solutions those which accomplish with the less power consumption. In Figs. 4.15 to 4.17 are depicted the solutions along generations.

NSGA_{SBX} with CBBA improves all objectives. Besides, the offset and power consumption exhibit slightly higher values, but all solutions accomplish target specifications, as shown in the left side of Table 4.8). The optimal solution without CBBA is found at generation 238, and with CBBA the optimal solution is found five generations faster. Also, with CBBA the number of solutions after 250 generations is higher (almost 4x), as shown in Figure 4.15(a).

The right side of Table 4.8 shows the solutions provided by NSGA_{DE}. With CBBA the optimal solution exhibits improvement for gain, PM, offset and power consumption. GBW has a lower value due to the gain increase, the noise values and ST are similar, only there is a slight decreasing in slew rate. With CBBA, the optimal solution is found 40 generations faster and the solutions increase (almost 3x) as shown in Figure 4.15(b).

The left side of Table 4.9 shows the solutions provided by $\text{MOEAD}_{\text{SBX}}$. With CBBA there is an improvement in gain, PM, slew rate, noise and power consumption. With CBBA the optimal solution is found more than 100 generations faster and there are almost 5x solutions more than $\text{MOEAD}_{\text{SBX}}$ without CBBA as shown in Figure 4.16(a). MOEAD_{DE} exhibits similar behavior. The optimal solution is found 30 generations faster and the number of solutions is 2x with CBBA (Figure 4.16(b)).

Table 4.10 lists the objective values for the optimal solutions for MOPSO. This time, CBBA achieves a general improvement for all the objective values, except for ST that exhibits a slight increment. The optimal solution is found more than 80 generations faster. Figure 4.17 shows an increase on the number of solutions after 250 generations (almost 5x).

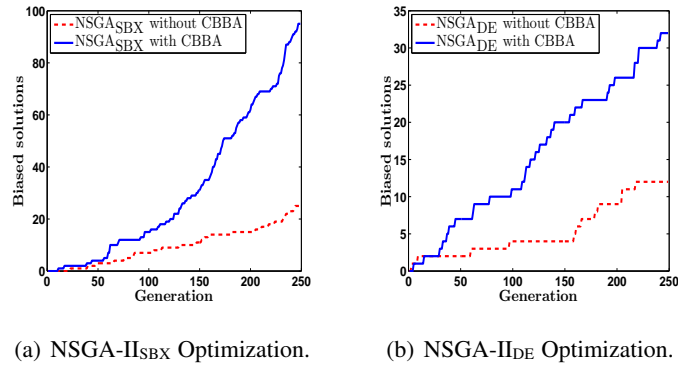


Figure 4.15: Solutions for the RFC OTA with and without CBBA for NSGA-II.

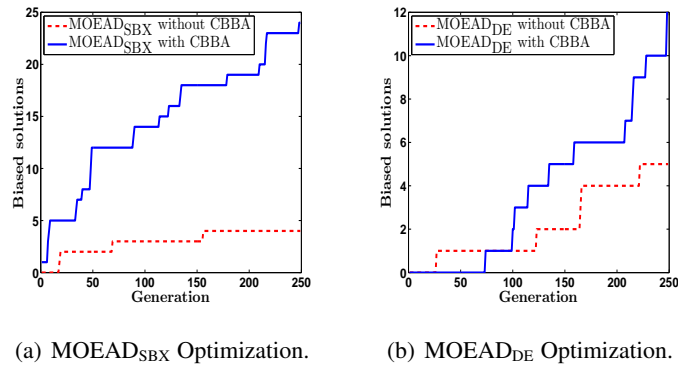


Figure 4.16: Solutions for the RFC OTA with and without CBBA for MOEAD.

Table 4.8: Optimal solutions for the RFC OTA with NSGA-II.

	Specs.	NSGA _{SBX}		NSGA _{DE}	
		Without CBBA	With CBBA	Without CBBA	With CBBA
Objectives					
Gain [dB]	≥ 55	65	61	61	64
GBW [MHz]	≥ 70	111	113	139	120
PM [deg]	≥ 65	62	68	65	75
Offset [<i>m</i> V]	≤ 11	0.15	0.30	0.51	0.28
ST [ns]	≤ 20	16.9	16.5	15.36	16
SR [V/ <i>μ</i> s]	≥ 48	80	120	97	87
Noise [<i>μ</i> V _{rms}]	≤ 69	70	69	68	68
PW [mW]	≤ 3.5	3.2	3.4	3.5	3.3
Variables					
<i>L</i> 1[<i>μ</i> <i>m</i>]	0.18	0.23	0.18	0.18	0.21
<i>L</i> 2[<i>μ</i> <i>m</i>]	0.5	0.24	0.18	0.19	0.18
<i>W</i> 1[<i>μ</i> <i>m</i>]	64	129.98	129.00	87.19	75.66
<i>W</i> 2[<i>μ</i> <i>m</i>]	32	126.00	82.15	127.12	78.39
<i>W</i> 3[<i>μ</i> <i>m</i>]	12	98.66	100.46	53.02	20.8
<i>W</i> 4[<i>μ</i> <i>m</i>]	4	41.49	47.77	17.45	72.42
<i>W</i> 5[<i>μ</i> <i>m</i>]	8	14.05	13.64	10.17	9.34
<i>W</i> 6[<i>μ</i> <i>m</i>]	32	65.3	82.06	128.53	66.84
<i>W</i> 7[<i>μ</i> <i>m</i>]	32	6.42	6.04	65.56	15.60
<i>W</i> 8[<i>μ</i> <i>m</i>]	4	7.03	2.96	29.58	46.91
Generation		238	233	166	126

4.7 Summary

A new current-branches-bias assignment (CBBA) approach has been introduced in order to accelerate the sizing process of an analog integrated circuit composed of MOSFETs. The ap-

Table 4.9: Optimal solutions for the RFC OTA with MOEAD.

	Specs.	MOEAD _{SBX}		MOEAD _{DE}	
		Without	With	Without	With
		CBBA	CBBA	CBBA	CBBA
Objectives					
Gain [dB]	≥ 55	62	63	63	66
GBW [MHz]	≥ 70	117	115	130	109
PM [deg]	≥ 65	50	69	58	69
Offset [<i>mV</i>]	≤ 11	0.12	0.19	0.19	0.20
ST [ns]	≤ 20	8.3	15.27	13.17	15.55
SR [V/ <i>μs</i>]	≥ 48	81	92	80	88
Noise [<i>μV</i> _{RMS}]	≤ 69	84	72	73	70
PW [mW]	≤ 3.5	3.4	3.3	3.5	3.1
Variables					
<i>L1</i> [<i>μm</i>]	0.18	0.58	0.21	0.18	0.21
<i>L2</i> [<i>μm</i>]	0.5	0.24	0.18	0.26	0.18
<i>W1</i> [<i>μm</i>]	64	111.60	120.95	80.51	75.66
<i>W2</i> [<i>μm</i>]	32	65.77	83.12	128.22	78.39
<i>W3</i> [<i>μm</i>]	12	77.47	88.59	111.68	20.8
<i>W4</i> [<i>μm</i>]	4	16.21	35.14	33.37	72.42
<i>W5</i> [<i>μm</i>]	8	13.36	13.65	9.41	9.34
<i>W6</i> [<i>μm</i>]	32	78.99	75.58	76.47	66.84
<i>W7</i> [<i>μm</i>]	32	28.42	6.65	8.52	15.60
<i>W8</i> [<i>μm</i>]	4	40.25	6.73	19.31	46.91
Generation		155	47	164	134

proach executes a recursive depth first search in the associated graph of the analog circuit modeled by using nullors. The approach determines current bias levels with the main goal to limit

Table 4.10: Optimal solutions for the RFC OTA with MOPSO.

	Specs.	MOPSO	
		Without CBBA	With CBBA
Objectives			
Gain [dB]	≥ 55	62	67
GBW [MHz]	≥ 70	132	134
PM [deg]	≥ 65	52	68
Offset [<i>m</i> V]	≤ 11	0.36	0.25
ST [ns]	≤ 20	11.9	13.1
SR [V/ μ s]	≥ 48	87	90
Noise [μ Vrms]	≤ 69	70	70
PW [mW]	≤ 3.5	3.43	3.39
Variables			
<i>L</i> 1[μ m]	0.18	0.18	0.27
<i>L</i> 2[μ m]	0.5	0.23	0.20
<i>W</i> 1[μ m]	64	70.08	130
<i>W</i> 2[μ m]	32	130	81.73
<i>W</i> 3[μ m]	12	130	20.8
<i>W</i> 4[μ m]	4	47.47	5.94
<i>W</i> 5[μ m]	8	7.74	15.72
<i>W</i> 6[μ m]	32	130	83.2
<i>W</i> 7[μ m]	32	17.34	61.21
<i>W</i> 8[μ m]	4	1.8	22.09s
Generation		122	39

the sizing search space in performing circuit optimization with multi-objective evolutionary algorithms, namely: NSGA-II, MOEAD and PSO. These algorithms were executed by using

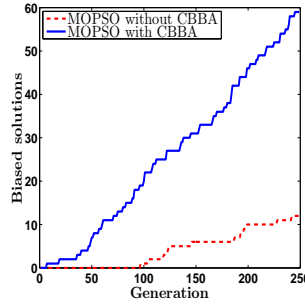


Figure 4.17: Solutions for the RFC OTA with and without CBBA for PSO.

SBX and DE as genetic operators.

The proposed CBBA approach was tested on two amplifiers: FC OTA and RFC OTA, both with different number of design variables and objectives. The results demonstrate the usefulness of the CBBA to accelerate the sizing process through a reduction in the number of generations needed to guarantee convergence and to generate feasible solutions while improving/preserving the circuit performances.

Our experiments show a reduction up to 100 generations to find an optimal solution and an increase up to 5x in the number of generated solutions by the evolutionary algorithms NSGA-II, MOEAD and MOPSO. As a result, our proposed CBBA approach can be used within any optimization process (deterministic or heuristic) to limit the search spaces for the different circuit variables, in order to accelerate the automatic sizing of analog integrated circuits.

Chapter 5

Circuit Variation Analysis

5.1 Introduction

This chapter is devoted to describe the variation analysis for analog circuits. It is exposed an outline of support variation methodologies and their classifications. It is presented a worst case approach by including sensitivity in the optimization of analog circuits by using the Richardson extrapolation. Finally, it is performed a non-worst case yield optimization by using an allocation budget simulations with the aim to reduce the runtime.

5.2 Variation Analysis for Analog Circuits

The analog design optimization involves not only finding nominal solutions to accomplish desired performances with constraints, but also it requires that such solutions guarantee robustness in front to the fabrication process [51, 100]. For instance, let $\mathbf{x}_i = (x_1^i, x_2^i)$ be a design that describes a given circuit in the variables search space. \mathbf{x}_i is subject to process variations and the design variables can lie into a “Tolerance Region” (R_{tol}), defined by the nominal values and the tolerances for each one of the design values. The process variation is depicted in Fig. 5.1 where each point represents a different design of the same circuit and R_{tol}^i is the tolerance region for \mathbf{x}_i .

Then \mathbf{R}_{tol} is the set of all the tolerance regions of the different designs. It is possible

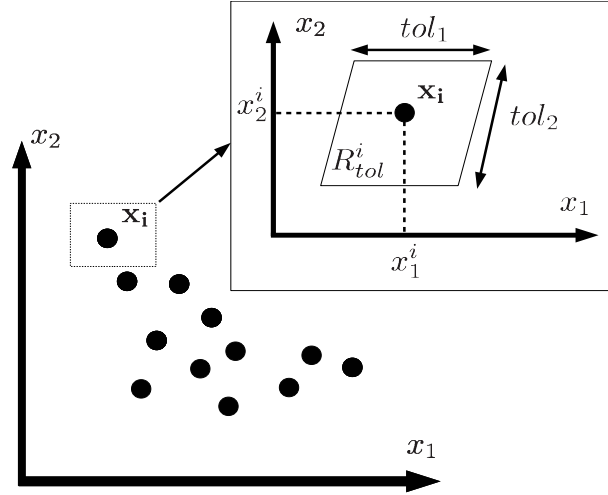


Figure 5.1: Example of process variation.

to define an “Acceptability Region” (\mathbf{R}_{ac}) which represents the region in the design variables space, within the performances and constraints specifications are accomplished. Having \mathbf{R}_{ac} and \mathbf{R}_{tol} exist two possibilities: the one is when $\mathbf{R}_{tol} \subseteq \mathbf{R}_{ac}$ (Fig. 5.2(a)) and the second is when $\mathbf{R}_{tol} \subset \mathbf{R}_{ac}$ (Fig. 5.2(b)) as shows Fig. 5.2 where is exposed a circuit with two design variables (x_1 and x_2), $\mathbf{R}_{ac}^* = \mathbf{R}_{ac} \cap \mathbf{R}_{tol}$ and $\mathbf{R}_{fail} = \mathbf{R}_{tol} - \mathbf{R}_{tol} \cap \mathbf{R}_{ac}$ is the “Failed Region” because in that region the specifications and constraints are not met (shaded region).

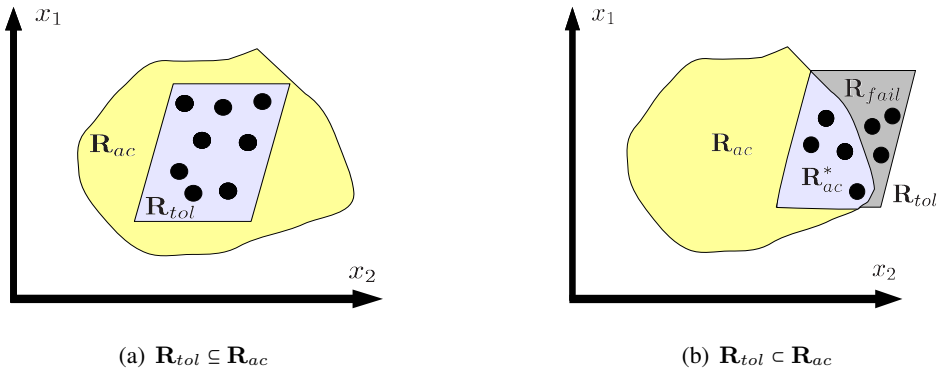


Figure 5.2: Acceptability and Tolerance Regions.

Due to the random behavior of the variations, for a given fabrication run it is possible to define the manufacturing yield (\mathbf{Y}) as the percentage of manufactured circuits which accom-

plish all the performance and constraints (acceptable circuits) while supporting process variations [51, 100, 101] as shown by (5.1). In this manner a yield value of 100% means that the tolerance region is completely inside the acceptable region (Fig. 5.2(a)) but usually a manufacture process exhibits less than 100% of the circuits fabricated preserving the performances and constraints specifications. Therefore, the yield is an statistical measure that allows to have an idea of the circuit robustness.

$$Y = \frac{|\mathbf{R}_{ac}^*|}{|\mathbf{R}_{tol}|} = \frac{\text{Number of acceptable circuits}}{\text{Number of fabricated circuits}} \quad (5.1)$$

$$\arg \max_{\mathbf{x} \in X, \mathbf{v} \in V} Y(\mathbf{x}, \mathbf{v}) \quad (5.2)$$

Then, the objective to include the support variation in the analog design aims to achieve the larger yield possible before fabricating the circuits. Herein it is possible to formulate the yield optimization problem as (5.2). Where \mathbf{x} is a “critical design” that accomplishes all the circuit constraints and within the variables space search X , and \mathbf{v} is a set of parameters due to process variations within the variation space V .

The support variation analysis, as shows Fig. 5.3 can be performed in several ways and can be grouped in: Worst Case Based and Non-Worst Case Based and this last can be grouped in sampling based or non-sampling based [100, 102].

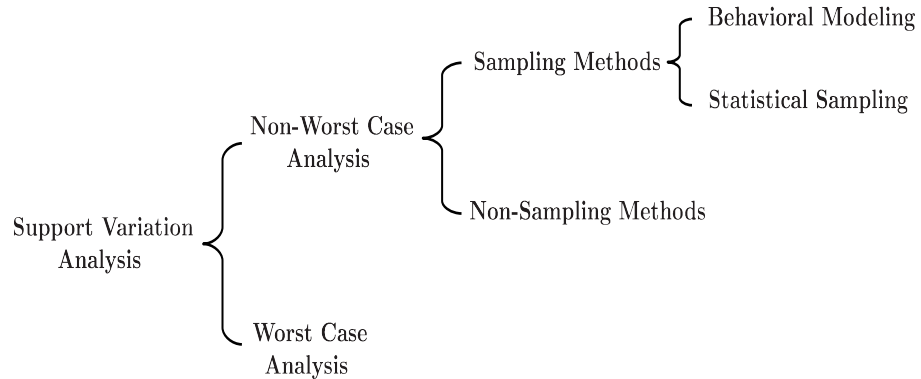


Figure 5.3: Classification of Support Variation Analysis [100].

The **Worst Case** based methodologies consist of identifying the different design values by

exploring the search space that produces extreme performances. This task can be performed with a vertex analysis as in WiCkeD [50] or as describes [100].

There exists a recent work in [103] where is performed a worst case bound analysis of analog circuits by including variations. In this case, linear models for active devices and the small signal transistor model are used to find transfer functions with variational coefficients for including global and local variations. There are provided two examples: a low-pass filter and an cascode op-amp, and this approach achieves an average speed up of 90x compared with a classical Monte Carlo approach.

The **Non-Worst Case** based methodologies are grouped in Sampling and Non-Sampling methods. Regarding to Non-Sampling methods it is possible to find works as in [102, 104] that are based on the sensitivities of the circuit performances relative to each design variable. In this case there are some issues to overcome to achieve that kind of analysis, as defining perturbation factors for each design variable and the sensitivity calculation of a circuit when there is not explicit equations that describe the performance as function of the design variables. Also this approach may be stalled when it is desired to find more accurate results, due to that with two slightly different design values the simulator finds the same performances result.

The sampling methods are grouped in: behavioral models based and statistical sampling based. The behavioral model based methodologies build regression models through successive simulations to estimate the yield, as in [105] where response surface methods are used and in [5] that exploits the Kriging model to approximate the yield. All these methodologies build the behavioral model by sampling and the number of samples needed is determined by the desired level of accuracy, while more accuracy more samples are needed. Also, the complexity of the models exhibit a strong dependence on the accuracy.

Finally, the statistical sampling based analysis, consist of calculating the yield from a large number of samples generated randomly over the search space of the design variables by simulating the fabrication process [100]. To perform the yield analysis, the circuit performances of each sample are calculated, and the number of samples that accomplish all the specifications and constraints divided by the total number of samples represents the yield. Once the factory provides the statistical transistor model, the only shortcoming is choosing the number

of samples for a given design because the accuracy of the yield depends on it.

From the shortcomings between Non-Worst and Worst case approaches we choose the last ones, because for a multi-objective optimization the Worst case approaches can be included easily. However, exists difference between the Sampling and the Non-Sampling Methods then the following sections show two Non-Worst Case approaches to broad the variation aware optimization of analog circuits: based on sensitivity and based on Monte Carlo simulations.

5.3 Sensitivity Analysis in the Multi-Objective Optimization of Analog Circuits by Applying Richardson Extrapolation

The feasible solutions provided by a multi-objective evolutionary algorithm (MOEA) in the optimal sizing of analog integrated circuits (ICs) can be very sensitive to process variations. Therefore, to select the optimal sizes of metal-oxide-semiconductor field-effect-transistors (MOSFETs) but with low sensitivities, we propose to perform multi-parameter sensitivity analysis. However, since MOEAs generate feasible solutions without an explicit equation, then we propose to apply numerical finite differences and Richardson extrapolation to approximate the partial derivatives associated to the sensitivities of the performances of an amplifier with respect to every MOSFET size. The proposed multi-parameter sensitivity analysis is verified through the optimization of a recycled folded cascode (RFC) operational transconductance amplifier (OTA). The results show that the optimal sizes of the MOSFETs, selected after executing the multi-parameter sensitivity approach, guarantee and improve the performances of the RFC OTA.

To have a general idea on analog integrated circuit sizing strategies developed by researchers and companies during the last 20 years, an overview on the classification and a brief description of the majority of them can be found in [23]. Although these works and other recently published strategies [13, 106] provide good sizing solutions, still the analog design community deals with the hard open problem related to process variations [52, 107]. In this manner, we propose to perform multi-parameter sensitivity analysis to the feasible solutions provided by a multi-objective evolutionary algorithm (MOEA), with the goal to select the optimal sizes of an analog IC but

with low sensitivities. Because very often, the best feasible solutions meeting extreme performance requirements are located at some peripherals of the feasible solution space, but some variability in the design parameters can transform a best solution to a worst one [52, 107–109]. Since our proposed multi-parameter sensitivity analysis is performed from numerical data instead of using explicit equations, we propose to apply numerical finite differences and Richardson extrapolation [110–113], to approximate the partial derivatives associated to the sensitivities of the sizing relationships W/L (width/large) of the MOSFETs. These processes are performed in two domains: variables W/L (design parameters) and objectives, where both are evaluated by linking HSPICE®.

The first step of our proposed approach consists on conventional optimization by applying the MOEA called non-dominated sorting genetic algorithm (NSGA-II) [76]. The second step is devoted to perform multi-parameter sensitivity analysis for all feasible solutions to discriminate those located in a delicate point which does not support the natural variations of the fabrication processes.

5.3.1 Multi-Parameter Sensitivity Analysis

The relative or normalized sensitivity (S) can be defined as the cause and effect relationship between the circuit elements variations, and the resulting changes in the performances response [114, 115]. Furthermore, in the design of analog ICs the lowest sensitivity is very desired.

Let $f_i(\mathbf{x})$ be an objective function (performance response), where $\mathbf{x} = [x_1, \dots, x_n]^T$ are the design variables. It is possible to relate small changes in the response of the performance (∂f_i , $i \in [1, m]$) to variations in the design variables (∂x_j , $j \in [1, n]$). It leads us to the single parameter sensitivity definition given by,

$$S_{x_j}^{f_i} \simeq \frac{x_j}{f_i} \frac{\partial f_i}{\partial x_j}. \quad (5.3)$$

According to (5.3), there is one sensitivity for every objective function in \mathbf{f} (see Eq. (5.11)) and for every variable in \mathbf{x} . Then, it is possible to define the multi-parameter sensitivity which sums the different single sensitivities regarding the different variables for every objective as

follows [115]:

$$S^{f_j} = \sqrt{\sum_{i=1}^n |S_{x_i}^{f_j}|^2 \cdot \sigma_{x_i}^2}, \quad (5.4)$$

where $S_{x_i}^{f_j}$ is calculated by (5.3), σ_{x_i} is a variability parameter of x_i and the square root is used to preserve the same units.

The performances of an analog IC are evaluated using HSPICE[®], and they are considered as the objective functions. As one can infer, using a numerical circuit simulator, there is not possibility to derive an explicit equation for every performance or objective function. Therefore, in order to calculate the partial derivative required by (5.3), the Richardson extrapolation described by (5.5), is used herein.

$$\frac{\partial f_i}{\partial x_j} \approx \frac{g_i(\mathbf{x}, j, d) - g_i(\mathbf{x}, j, -d)}{2d}, \quad \text{with } d \rightarrow 0 \quad (5.5)$$

where function g_i is defined as:

$$\begin{aligned} g_i : \mathbb{R}^n &\rightarrow \mathbb{R} \\ g_i(\mathbf{x}, j, d) &= f_i(\mathbf{y}) \mid y_k = x_k \text{ for } k \neq j \text{ and } y_j = x_j + d \end{aligned} \quad (5.6)$$

In (5.5), d is a step parameter that is updated in each iteration [116], for this case $d = 2^{-u}d_{u-1}$, d_0 is assigned to an initial value and u is the current iteration. The recursive calculation continues until a tolerance error, as stopping criterion (δ), is reached.

Our proposed multi-parameter sensitivity analysis approach is based on the Richardson extrapolation sketched in Algorithm 13. This algorithm is performed until the stopping criterion or a maximum number of iterations, is reached. The Richardson extrapolation is a sequence acceleration method used to improve the rate of convergence of calculation of the partial derivatives.

Due to the iterative processes, it is possible to have stagnation in the $f'_{u,v}$ evaluation (line 9 of Algorithm 13), when the very small value of d does not produce difference between $f'_{u,v-1}$ and $f'_{u-1,v-1}$. If that happens, the algorithm save the last value before the stagnation to avoid a wrong derivative value.

For instance, let $\mathbf{x} = [x_1, x_2, x_3]^T$ be any solution from the feasible solutions set and $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x})]^T$ its objective vector. For the objective function f_1 , and for a

Algorithm 13 Richardson Extrapolation

```

1:  $h = h_0$ 
2: for  $u = 0; u < \text{MaxLoops} ; u++$  do
3:   for  $v = 0; v < \text{MaxLoops} ; v++$  do
4:     if  $v == 0$  then
5:        $g^+ =$  Function evaluation with the parameter  $+ d$ 
6:        $g^- =$  Function evaluation with the parameter  $- d$ 
7:        $f'_{u,v} = (g^+ - g^-)/(2 * d)$ 
8:     else
9:        $f'_{u,v} = f'_{u,v-1} + (f'_{u,v-1} - f'_{u-1,v-1})/(2 * (2 * v) - 1)$ 
10:    if  $|f'_{u,v} - f'_{u,v-1}| < \delta$  then
11:      break
12:    end if
13:  end for
14:   $d \leftarrow d/2;$ 
15: end for
16: end for
17: return  $f'_{u,v}$ 

```

given initial value of d , the first estimations for the partial derivatives

$$\begin{aligned} \frac{\partial f_1}{\partial x_1} &\approx \frac{g1(\mathbf{x}, 1, d) - g1(\mathbf{x}, 1, -d)}{2d} \\ \frac{\partial f_1}{\partial x_2} &\approx \frac{g1(\mathbf{x}, 2, d) - g1(\mathbf{x}, 2, -d)}{2d} \\ \frac{\partial f_1}{\partial x_3} &\approx \frac{g1(\mathbf{x}, 3, d) - g1(\mathbf{x}, 3, -d)}{2d} \end{aligned}$$

are calculated. Next, the Richardson extrapolation is executed. That way, the multi-parameter sensitivity for function f_1 is calculated as

$$S^{f_1} = \sqrt{(S_{x_1}^{f_1})^2 \sigma_{x_1}^2 + (S_{x_2}^{f_1})^2 \sigma_{x_2}^2 + (S_{x_3}^{f_1})^2 \sigma_{x_3}^2}$$

The other multi-parameter sensitivities S^{f_2} and S^{f_3} are calculated with the same procedure.

In our implementation of the Richardson extrapolation, the three partial derivatives for all the

functions f_1 , f_2 and f_3 , with respect to a variable x_i (say x_1), are calculated in a single HSPICE simulation, because those values correspond to the variation of $x_1 \pm d$. The Richardson extrapolation is executed until the stop criteria or the given maximum number the iterations is reached for all the three partial derivatives.

With the aim to highlight the behavior of the Richardson extrapolation, an example on the calculation of the multi-parameter sensitivity of the SRN function described by (5.7) [76], is exposed herein. It consists to evaluate two objective functions $[f_1(\mathbf{x}), f_2(\mathbf{x})]^T$, with two variables $\mathbf{x} = [x_1, x_2]^T$, and taking into account two constraints h_1 and h_2 . This optimization problem is minimized by using the NSGA-II algorithm, leading to the result shown in Figure 5.4. From these numerical feasible solutions we apply the Richardson extrapolation to evaluate the sensitivity of every solution, with $d_0 = 5\%$ for the current variable and $\delta < 1\%$.

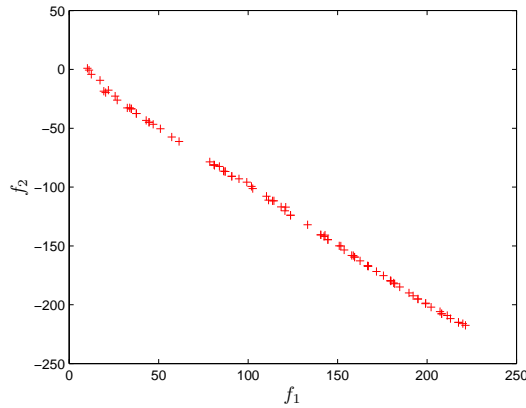
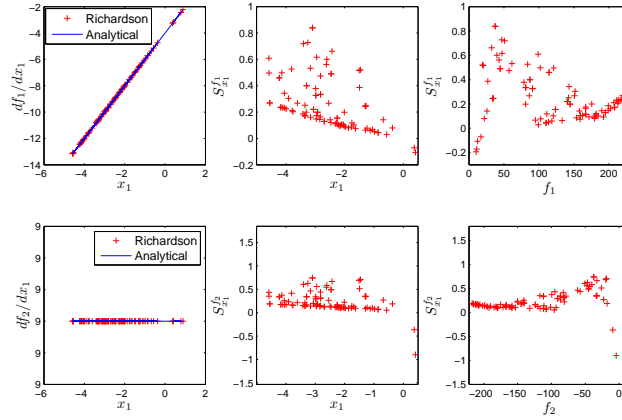


Figure 5.4: Optimization of SRN by applying NSGA-II.

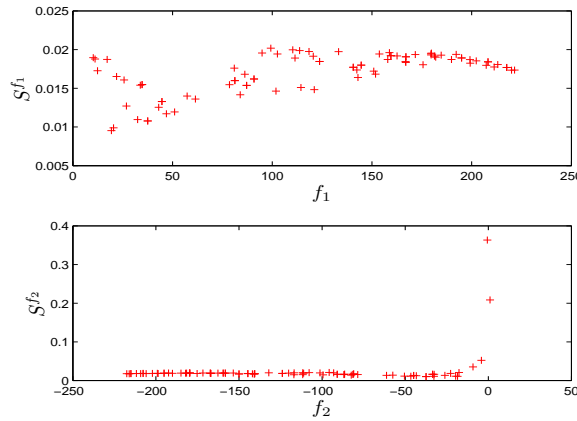
$$SRN = \begin{cases} f_1(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 1)^2 + 2 \\ f_2(\mathbf{x}) = 9x_1 - (x_2 - 1)^2 \\ h_1(\mathbf{x}) = x_1^2 + x_2^2 \leq 225 \\ h_2(\mathbf{x}) = x_1^2 - 3x_2^2 \leq -10 \end{cases} \quad (5.7)$$

Figure 5.5 shows the analytical and numerical solution by applying Richardson extrapolation of the partial derivatives for f_1 and f_2 for the variable x_1 : $\partial f_1 / \partial x_1 = 2(x_1 - 2)$ and $\partial f_2 / \partial x_1 = 9$. These results demonstrate that the Richardson extrapolation calculated numeri-

Figure 5.5: Sensitivity of SRN with respect to x_1 .

cally, agrees with the result by evaluating (5.7) analytically.

If we focus on the sensitivity analysis, Figure 5.5 shows that for f_1 the closer to zero values of x_1 exhibit low sensitivity, and for f_2 the closer to zero values exhibit the lowest sensitivity too. Figure 5.6 depicts the multi-parameter sensitivity evaluated from (5.4), for f_1 and f_2 with $\sigma = 1\%$ (as variables represent sizes in circuit optimization, we consider the same variability value for them); now it can be noted that few values of both functions exhibit low sensitivity.

Figure 5.6: Multi-parameter sensitivity of SRN function for f_1 and f_2 .

Finally, from the feasible solution set it is possible to chose the multi-parameter sensitivity lower than 0.015, considering both objective functions. As a result, the feasible solutions with

low sensitivities are shown in Figure 5.7. They were selected from the feasible solutions in Figure 5.4. This example demonstrates the usefulness of the Richardson extrapolation to perform multi-parameter sensitivity analysis from numerical data.

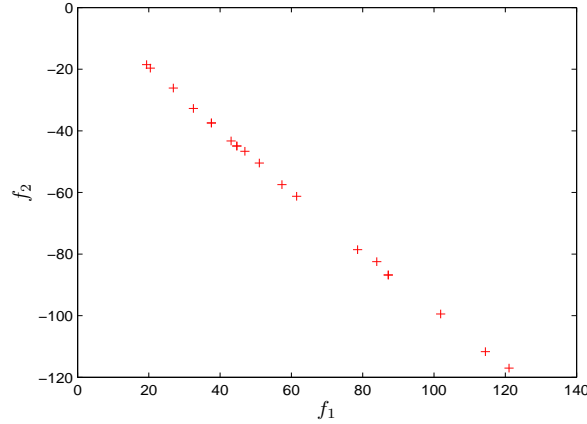


Figure 5.7: Feasible solutions for SRN after applying multi-parameter sensitivity analysis.

5.3.2 Proposed Optimization System Including Multi-Parameter Sensitivity Analysis

Our proposed approach to select optimal sizes with low sensitivities has been programmed using MATLAB[®], and the circuit under optimization is simulated with HSPICE[®] through successive simulations [72]. The optimization of the circuit performances is done by modifying the width (W) and length (L) of the MOSFETs.

In Figure 5.8, it can be appreciated that our proposed optimization approach is divided into two general stages: initialization and optimization. In the initialization stage the parameters as maximum number of generations, population size and sensitivity parameters (d_0 and δ), are declared. In the second stage, the NSGA-II algorithm is applied to generate feasible solution sets. In this stage HSPICE[®] is linked to evaluate the objective functions and constraints. Only the solutions that meet the specifications (constraints) are introduced to the multi-parameter sensitivity analysis based on the Richardson extrapolation. Afterwards, the non-dominated sort is performed giving priority to the solutions with a measure of multi-parameter sensitivity be-

cause are the solutions that accomplish with the target specifications and the constraints. The final solution set contains solutions with the less multi-parameter sensitivity.

The efficiency of the procedure depends on the efficiency of both, the NSGA-II and the multi-parameter sensitivity calculation. Regarding to NSGA-II, its efficiency is $O(N^2M)$, where N is the number of individuals and M is the number of objectives. The multi-parameter sensitivity calculation efficiency is $O(\psi n M N_v)$, where ψ is the number of iterations in the Richardson extrapolation, n is the number of variables and M is the number of objectives and N_v is the number of solutions that accomplishes the constraints then $N_v \leq N$. The worst case for ψ is when is achieved the *MaxLoop* value (Algorithm 13) and when $N_v = N$. These efficiencies indirectly depend on the simulator efficiency due that both, the optimization and the multi-parameter sensitivity evaluation are performed from a circuit simulation.

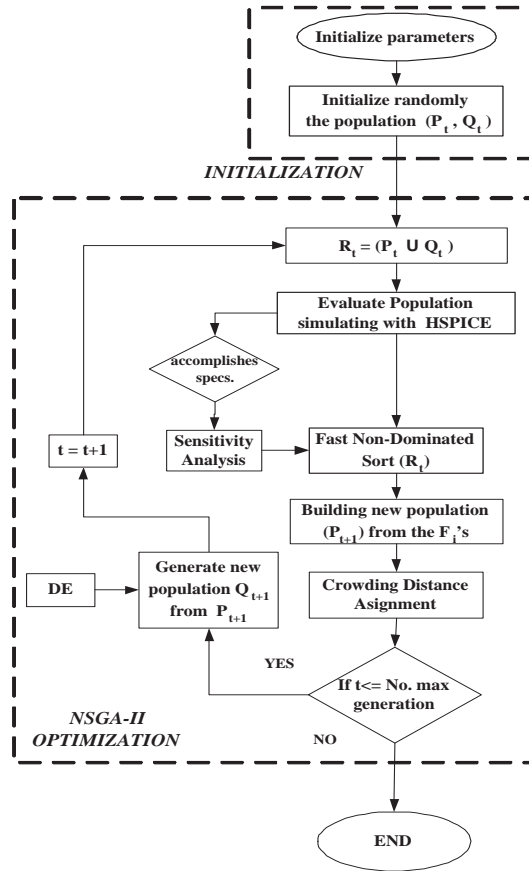


Figure 5.8: Flow Diagram.

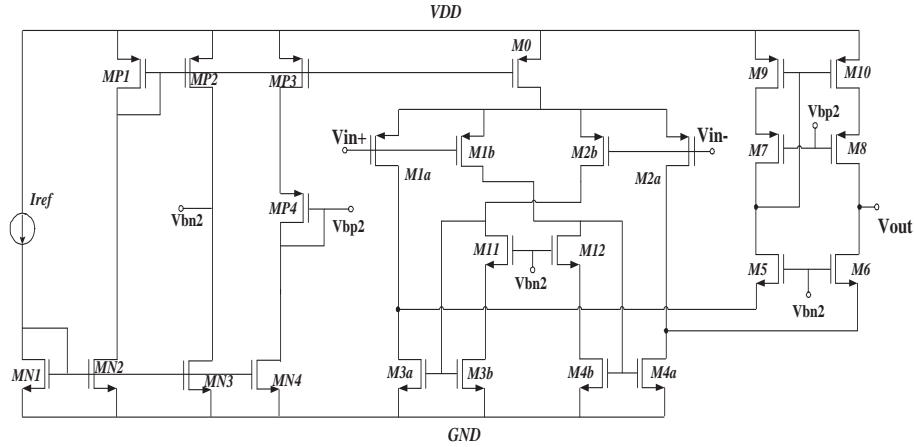


Figure 5.9: Recycled Folded Cascode OTA.

5.3.3 Example

The proposed optimization approach including multi-parameter sensitivity analysis is tested on the recycled folded cascode (RFC) operational transconductance amplifier (OTA) shown in Figure 5.9. It was taken from [99], but now we include the design of the biasing circuitry shown in the left part. The biasing circuitry consists of $\{MP1, \dots, MP4\}$ and $\{MN1, \dots, MN4\}$, to provide two voltages: V_{bp} and V_{bn} .

The optimization is executed to accomplish the eight objectives already provided in [99]: DC gain, gain bandwidth product (GBW), phase margin (PM), input referred noise, input offset, settling time (ST), slew rate (SR) and power consumption (PW). This circuit is encoded with ten variables (design parameters) for the MOSFETs, as shown in Table 5.1.

$$\begin{aligned} \mathbf{f}(\mathbf{x}) &= [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_8(\mathbf{x})]^T \\ &= \left[\frac{1}{\text{Gain}}, \frac{1}{\text{GBW}}, \frac{1}{\text{PM}}, \text{Noise}, \text{Offset}, \text{ST}, \frac{1}{\text{SR}}, \text{PW} \right]^T \end{aligned} \quad (5.8)$$

The optimization problem for this circuit is expressed as in (5.11), with $m = 8$, $n = 10$ and $p = 33$, where $\mathbf{f}(\mathbf{x})$ is the vector formed by the eight objectives, accommodated as shown in (5.8), to deal with a minimization optimization problem. Therefore, the objectives Gain, GBW, PM and SR have been inverted. X is the search space for the variables listed in Table 5.1, and the decision space for $\mathbf{x} = [x_1, \dots, x_{10}]^T$. However, the variables x_1 and x_2 have

Table 5.1: Encoding for the RFC OTA shown in Figure 5.9.

<i>gene</i>	Design Variable	Encoding Transistors
x_1	L_1	M0,M3a,M3b,M4a,M4b,M9,M10
		MN1, ..., MN4, MP1, ..., MP4
x_2	L_2	M5, ..., M8
	$2 \cdot L_2$	M1a,M1b,M2a,M2b
x_3	W_1	M0, MP1
x_4	W_2	M1a,M1b,M2a,M2b
x_5	W_3	M3a,M4a
x_6	W_4	M3b,M4b
x_7	W_5	M5, M6,MN3,MN4
	$2 \cdot W_5$	MN1,MN2,MP4
	$4 \cdot W_5$	MP2,MP3
x_8	W_6	M7, M8
x_9	W_7	M9, M10
x_{10}	W_8	M11, M12

the range $[0.18 \dots 0.72]$ (in μm). The rest of the variables have the range $[0.18 \dots 140]$ (in μm). Finally, in a first experiment $h_k(\mathbf{x})$, $k = 1, \dots, 33$ are performance constraints. There are included as constraints the saturation condition in all the 25 transistors and the target specifications for the eight objectives. In a second experiment which includes a multi-parameter sensitivity, $h_k(\mathbf{x})$, $k = 1, \dots, 41$ are the above constraints plus eight multi-parameter sensitivities. The optimization for the first experiment was performed along 250 generations over 4 runs with a population size of 250. For the second experiment, the optimization is stopped after several generations with the same multi-parameter sensitivity because the optimization takes more time than the first experiment. For DE there were arbitrarily selected $\gamma=1$ and $\eta=0.4$ for both experiments.

The RFC OTA is biased with $I_{ref} = 400\mu A$ and $V_{DD} = 1.8V$. The electrical measurements were executed with a load capacitor of 5.6pF and the HSPICE® simulations were performed

with a LEVEL 49 standard CMOS Technology of $0.18 \mu m$. The parameters for the sensitivity analysis are $d_0 = 3\%$ for the design values, and $\delta = 3\%$. The aim to use percentage values for d_0 and δ , is the possibility to manage different design values with different magnitudes. In this example, σ_i is proposed to be 3% .

The size of the final solution set is around 60, in average for all the runs in this experiment. Table 5.2 shows the target specifications (Specs.), minimum, maximum, average and standard deviation for all the objective functions among the final solution set, however those results do not take into account the sensitivities of every feasible solution. The target specifications to be improved are the values of the objective functions or performances evaluated with the sizes already published in [99]. The application of our optimization stage provides better performances compared to [99], for every objective function. These results are highlighted with bold font. From Table 5.2 one can gain an insight on what to expect for this RFC OTA circuit topology working under these design conditions. Later on, this table is an important base line to compare the results that the multi-parameter sensitivity analysis will generate.

The optimization works with a multi-objective problem, then the best performances for the eight objective functions are listed in Table 5.3, where \mathbf{x}_1 is the best solution for gain, \mathbf{x}_2 is the best solution for GBW and so on with PM, Noise, Offset, ST, SR and PW. For instance, the maximum gain (solution \mathbf{x}_1) is 68 dB; this best point, has GBW=107.97 MHz, PM=76.86 deg, Noise=52.62 μV_{rms} , Offset=20.04 μV , ST= 17.10 nsecs, SR= 88.04 V/ μsec and PW= 3.25 mWatts. This optimum gain is achieved with $L_1=0.36 \mu m$, $L_2=0.18 \mu m$, $W_1=132.48 \mu m$, $W_2=58.5 \mu m$, $W_3=18.9 \mu m$, $W_4=9.9 \mu m$, $W_5=16.2 \mu m$, $W_6=32.76 \mu m$, $W_7=5.22 \mu m$ and $W_8=24.3 \mu m$.

The next step consisted of performing an optimization including a multi-parameter sensitivity analysis. In such way that the sensitivity adds eight constraints more: the multi-parameter sensitivities for each one of the eight objectives. Then we have in total 49 constraints for this second experiment, the firsts 33 constraints accomplish with the specifications and the next eight accomplish with the less multi-parameter sensitivity.

In this experiment the size of the final solution set is around the population size, although that within such set, all the solutions accomplish the target specifications, each one has a differ-

Table 5.2: Best points for the RFC OTA without sensitivity analysis.

Objective	Specs.	MAX	MIN	AVG	STD
Gain [dB]	> 65.35	68.00	66.03	67.44	0.37
GBW [MHz]	> 89.57	123.14	96.80	105.15	5.19
PM [deg]	> 75.47	79.45	75.47	76.76	0.74
Noise [μ Vrms]	< 68.41	69.42	42.98	58.63	5.51
Offset [μ V]	< 206.79	99.90	0.00	50.33	27.11
ST [ns]	< 20.14	18.50	15.29	17.33	0.70
SR [V/ μ s]	> 76.99	121.11	77.00	81.38	4.92
PW [mW]	< 3.09	3.30	3.04	3.22	0.06

Table 5.3: Best sizing solutions without sensitivity analysis for the RFC OTA.

	Specs.	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
L_1 [μm]	0.5	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36
L_2 [μm]	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18
W_1 [μm]	64	132.48	85.32	85.86	130.14	78.3	85.32	178.56	57.78
W_2 [μm]	32	58.5	72	41.94	68.04	49.68	72	49.5	58.14
W_3 [μm]	12	18.9	14.94	17.64	3.96	9.9	14.94	21.24	17.28
W_4 [μm]	4	9.9	7.02	8.1	1.98	4.86	7.02	12.42	6.66
W_5 [μm]	8	16.2	10.62	10.8	16.02	10.08	10.62	21.6	7.2
W_6 [μm]	32	32.76	18	17.1	26.64	69.66	18	34.2	52.92
W_7 [μm]	32	5.22	7.74	8.28	5.94	5.22	7.74	4.5	5.58
W_8 [μm]	32	24.3	10.62	18	9	31.14	10.62	1.8	3.96
Gain [dB]	> 65.35	68.00	67.57	66.13	66.86	67.79	67.57	67.97	67.69
GBW [MHz]	> 89.57	107.97	123.14	102.47	116.12	102.79	123.14	102.04	100.04
PM [deg]	> 75.47	76.86	75.57	79.45	75.47	77.14	75.57	76.99	75.50
Noise [μ Vrms]	< 68.41	52.62	49.23	58.60	42.98	51.92	49.23	54.96	61.54
Offset [μ V]	< 206.79	20.04	60.14	75.42	37.17	0.00	60.14	9.75	5.76
ST [ns]	< 20.14	17.10	15.29	18.08	16.22	17.51	15.29	17.78	18.08
SR [V/ μ s]	> 76.99	88.04	77.96	78.93	78.31	79.66	77.96	121.11	79.27
PW [mW]	< 3.09	3.25	3.29	3.30	3.28	3.25	3.29	3.25	3.04

ent value of multi-parameter sensitivity, then it is possible to chose the solutions with the lowest one. By selecting the lowest five solutions in each run, the Table 5.4 shows the target specifications (Specs.), minimum, maximum, average and standard deviation for all the objective functions including multi-parameter sensitivity in the optimization for every feasible solution. As before, the target specifications to be improved are the values of the objective functions or performances evaluated with the sizes already published in [99]. In this second experiment, it is proposed selecting the feasible solutions with the lowest multi-parameter sensitivity, in this case it is necessary to sacrifice some of the objectives with the aim to preserve the best values of the remaining ones. Then it was decided to allow slightly higher values of power consumption.

The application of our optimization stage provides better performances compared to [99], for every objective function except for power consumption which is slightly above the target specification. As before, the best results are highlighted with bold font. By comparing Tables 5.2 and 5.4, it is possible to see how the best results from the first experiment were lost, but nevertheless the best results for the second experiment still improve the targets except for the power consumption. Gain, PM and PW are almost in the same value than the first experiment. GBW, Noise and ST are closer to the first experiment values and finally, SR and Offset were decreased significantly compared with the first experiment, but still they are better than targets specs.

The best performances for the eight objective functions in the second experiment are listed in Table 5.5, where \mathbf{x}_1 is the best solution for gain, \mathbf{x}_2 is the best solution for GBW and so on with PM, Noise, Offset, ST, SR and PW. In this case, the solution for gain, GBW, PM, Noise, ST and PW is the same ($\mathbf{x}_1 = \mathbf{x}_2 = \mathbf{x}_3 = \mathbf{x}_4 = \mathbf{x}_6 = \mathbf{x}_8$).

At the beginning of the optimization including multi-parameter sensitivity, there are taking into account only the constraints that corresponding to the saturation condition of transistors and targets specifications. As soon as a solution accomplishes those constraints, there are included the multi-parameter sensitivity constraints, allowing categorize those solutions with the less multi-parameter sensitivity among the best ones, to guide the optimization. During this process, it is possible to see how the multi-parameter sensitivity is reduced from the moment when are found solutions which accomplish the first constraints. When the multi-parameter sensitivity

Table 5.4: Best points for the RFC OTA including sensitivity analysis.

Objective	Specs.	MAX	MIN	AVG	STD
Gain [dB]	> 65.35	67.83	66.46	67.01	0.42
GBW [MHz]	> 89.57	106.52	94.63	96.72	2.97
PM [deg]	> 75.47	77.30	75.48	75.96	0.45
Noise [μ Vrms]	< 68.41	66.40	55.27	63.51	3.27
Offset [μ V]	< 206.79	96.97	0.03	34.84	31.96
ST [ns]	< 20.14	18.49	16.90	18.25	0.36
SR [V/ μ s]	> 76.99	79.64	77.09	77.63	0.56
PW [mW]	< 3.09	3.30	3.24	3.28	0.02

Table 5.5: Best sizing solutions including sensitivity analysis for the RFC OTA.

	Specs.	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
L_1 [μm]	0.5	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36
L_2 [μm]	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18
W_1 [μm]	64	125.46	120.06	132.12	132.12	152.82	120.06	125.46	132.12
W_2 [μm]	32	45.18	52.2	54.18	54.18	45.72	52.2	45.18	54.18
W_3 [μm]	12	38.52	42.3	25.56	25.56	30.96	42.3	38.52	25.56
W_4 [μm]	4	19.62	22.14	14.22	14.22	16.92	22.14	19.62	14.22
W_5 [μm]	8	15.66	14.76	16.2	16.2	18.72	14.76	15.66	16.2
W_6 [μm]	32	61.92	25.92	18.72	18.72	59.58	25.92	61.92	18.72
W_7 [μm]	32	8.82	6.84	5.22	5.22	4.5	6.84	8.82	5.22
W_8 [μm]	32	8.28	2.88	3.06	3.06	12.96	2.88	8.28	3.06
Gain [dB]	> 65.35	67.83	67.40	67.63	67.63	67.45	67.40	67.83	67.63
GBW [MHz]	> 89.57	100.06	106.52	104.52	104.52	96.01	106.52	100.06	104.52
PM [deg]	> 75.47	76.37	75.55	77.30	77.30	76.55	75.55	76.37	77.30
Noise [μ Vrms]	< 68.41	61.86	59.07	55.27	55.27	59.64	59.07	61.86	55.27
Offset [μ V]	< 206.79	96.97	60.25	30.83	30.83	0.03	60.25	96.97	30.83
ST [ns]	< 20.14	18.13	16.90	17.52	17.52	18.40	16.90	18.13	17.52
SR [V/ μ s]	> 76.99	79.64	78.32	77.37	77.37	77.87	78.32	79.64	77.37
PW [mW]	< 3.09	3.29	3.27	3.24	3.24	3.25	3.27	3.29	3.24

is taken into account, is reduced over the next generations, until it is reached the lowest value over several generations. Figure 5.10 shows this behavior, in each one of the four runs, and it is possible to see how, for all the runs, after the generation 40 began to appear solutions with multi-parameter sensitivity and before the generation 100 reaches its lowest value.

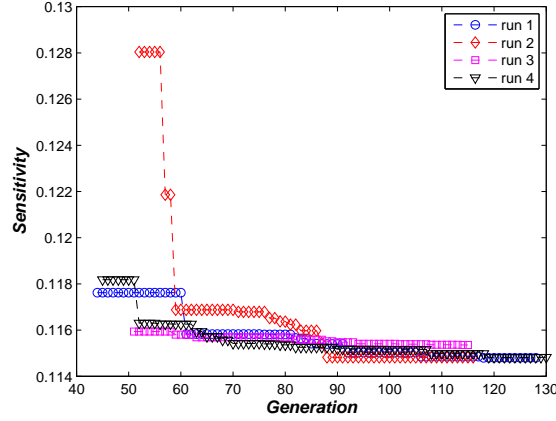


Figure 5.10: Behavior of Multi-Parameter Sensitivity vs Generations.

5.4 Optimal Computing Budget Allocation in the Yield Optimization of Analog Circuits

The work in the previous section is based on sensitivities to design variables, but a more realistic model of the problem is calculating yield based on the process variables distribution. The statistical methods are based on design of experiments procedures to sampling the search space in a random manner with the aim to emulate the actual fabrication process. The main idea is to generate design values according with probability density functions, simulate them and verifying the performances and constraints specifications. In this case, while larger the number of samples can be obtained more accuracy then exist works as [52] where a Latin Hypercube Sampling is performed to reduce the number of samples needed for 99% of yield with a minimal error. This methodology can be performed easily compared with the rest and it is completely circuit performances, transistor model and technology independent. Despite since 90's [100], the statistical based methods have been considered more reliable and more trustworthy than

other methodologies, the computational effort to perform the statistical methodologies is high due to the fact that each design needs to be simulated several times including the different variations.

In this manner, we applied a budget allocation methodology on the yield multi-objective optimization of analog circuits with the aim to reduce the time required for a classical yield optimization based on statistical simulations.

5.4.1 Optimal Computing Budget Allocation (OCBA)

The main task in the analysis variation is to identify the best designs when the variables of the circuit (\mathbf{x}) are under variations (\mathbf{v}). By setting a budget simulations there exist several ways to distribute the simulations among the different designs. An Optimal Computing Budget Allocation (OCBA) [117] is a strategy that distributes the large portion of the budget simulations among the critical designs and limits the simulations for the non-critical designs with the aim to enhance the efficiency of the optimization in simulation experiments. The final result of applying an OCBA is a reduction in the total simulation cost.

Among the OCBA's there exists one in [118] (CCY) that achieves a 74% reduction in the computation time. CCY has shown a reduction time of one order compared with the basic Monte Carlo for mono-optimization yield [52]. By setting ψ_T as the total budget simulations for the whole optimization and there are N_v critical designs. Equation (5.2) now is defined as (5.9), where ψ_i is the number of simulations performed for the i -th design.

$$\arg \max_{\mathbf{x} \in X, \mathbf{v} \in V} Y(\mathbf{x}, \mathbf{v}) \quad (5.9)$$

$$\text{subject to } \psi_T = \sum_i^{N_v} \psi_i$$

CCY proposes that from a total budget of simulations ψ_T to be distributed among N_v designs the yield problem can be maximized when (5.10) is accomplished, where b denotes the design having the best yield, σ_i is the variance of the i -th critical solution and $\delta_{b,i} = Y_b(\mathbf{x}, \mathbf{v}) - Y_i(\mathbf{x}, \mathbf{v})$ is the difference between the b -th and i -th yields. Such equations maximize the probability that design b is actually the best design among N_v designs [118].

$$\frac{\psi_i}{\psi_j} = \left(\frac{\sigma_i / \delta_{b,i}}{\sigma_j / \delta_{b,j}} \right)^2, i, j \in 1, 2, \dots, N_v \quad (5.10a)$$

$$\psi_b = \sigma_b \sqrt{\sum_{i=1, i \neq b}^{N_v} \frac{\psi_i^2}{\sigma_i^2}} \quad (5.10b)$$

Algorithm 14 OCBA Pseudocode

```

1:  $t = 0$ .
2: for  $i = 1, \dots, N_v$  do
3:    $\psi_i = \psi_0$ 
4:   Perform  $\psi_i$  simulations for the  $i$ -th design
5:   Update  $\sigma_i$  and  $\delta_{b,i}$ 
6: end for
7: repeat
8:   for  $i = 1, \dots, N_v$  do
9:     Increase  $\psi_i$  by  $\Delta$  according with (5.10)  $\rightarrow \psi_i^{t+1} = \psi_i^t + \Delta$ 
10:    Perform  $\psi_i^{t+1} - \psi_i^t$  simulations
11:    Update  $\sigma_i$  and  $\delta_{b,i}$ 
12:   end for
13:    $t+ = 1$ .
14: until  $\sum_{i=1}^t \psi_i^t \geq \psi_T$ 

```

Algorithm 14 depicts a sequential algorithm for OCBA, where t is the number of iteration, ψ_0 is the initial number of simulations for all the solutions and Δ is the increment of Ψ (number of simulations). From [118] it is proposed $5 \leq n_0 \leq 20$ and $5 \leq \Delta \leq 0.1 \cdot N_v$. Through the lines 1 to 6 of Algorithm 14 is performed an initialization process by assigning ψ_0 simulations for all the designs. The line 7 starts a loop which stops until the total budget is achieved ($\sum_{i=1}^t \psi_i^t \geq \psi_T$). From line 8 to 12, there is a loop that calculates the new simulations budget allocation for each design according with (5.10) and the new budget simulations are performed to update the mean (expressed by δ_i) and the standard deviation (σ_i) of each design.

The OCBA approach, besides by taking into account the accuracy of the statistical samples (represented by the mean), estimates the number of simulations of each design according to the

precision of the samples (represented by the standard deviation). In this manner there exists a trade-off between accuracy and precision to allocate optimally a simulations budget allowing improving the efficiency of the yield computation.

5.4.2 Proposed Optimization System Including Yield Analysis by Using OCBA

The proposed optimization system has been programmed using PERL 5.12 and the circuit simulations are performed with HSPICE®. The optimization of the circuit performances is done by modifying the values of some circuit parameters such as voltages, currents, width (W) and length (L) of the MOSFETs, among others. For the yield analysis it is required the inter-die and intra-die parameters provided by the factory.

In Fig. 5.11 is depicted the flow diagram based on Fig. 3.1 but this time the yield analysis is included and it can be performed with any method. In this work the yield analysis will be compared by using a classical Monte Carlo analysis and the OCBA analysis. In the circuit evaluation stage, HSPICE® is linked to evaluate the objective functions and constraints. Only the solutions that meet the specifications are introduced to the yield analysis.

To perform the yield analysis is required the SWEEP MONTE instruction provided by the simulator. For the classical Monte Carlo the number of iterations is fixed and for OCBA analysis the number of simulations is calculated in each iteration. Finally, the last generation has the solutions with the highest yield. In Appendix A it is exposed an embedded software tool GUI developed with all this optimization methodology.

The efficiency of the procedure depends on the efficiency of both, the evolutionary algorithm and the yield analysis along all the optimization process. Regarding to the evolutionary algorithm, its efficiency is proportional to the number of individuals or population size (N) and the number of objectives M . The yield analysis efficiency is $O(\psi N_v)$, where ψ_t is the total simulations budget and N_v is the number of solutions that accomplish the constraints therefore $N_v \leq N$, and the worst case is when $N_v = N$. These efficiencies indirectly depend on the simulator efficiency due that both, the optimization and the yield analysis are performed by circuit simulations.

Regarding to the design constraints, they are grouped in strong and weak constraints as

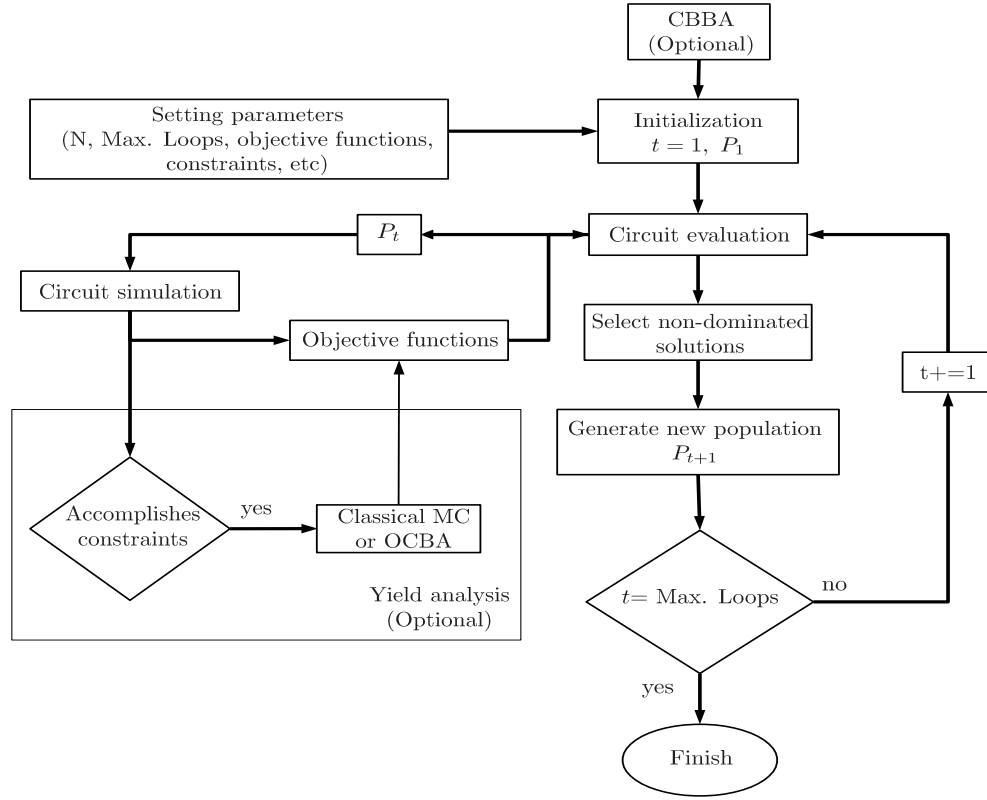


Figure 5.11: Flow Diagram for Optimization including OCBA.

in [28], but this time before and after the yield analysis. In this manner, the Multi-Objective Problem formulation described in (5.11), now is redefined as :

$$\begin{aligned}
 &\text{minimize} && \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_M(\mathbf{x}))^T \\
 &\text{subject to} && h(\mathbf{x}) = 0 \\
 &\text{where} && \mathbf{x} \in X \\
 &&& h(\mathbf{x}) = \sum_{l=0}^p \varrho(h_l)
 \end{aligned} \tag{5.11}$$

where $X \subset \mathbb{R}^n$ is the decision space for the design variables, $\mathbf{x} = (x_1, \dots, x_n)$ is called the decision vector. $\mathbf{f}(\mathbf{x})$ is the performance objective vector, $f_j(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$, $m = 1 \dots M$ ($M \geq 2$) are performance objective functions and $h_l(\mathbf{x})$, $l = 1 \dots p$, are performance constraints. The $\varrho(h_l)$ function is defined in (5.12) and the $\{\varrho^{sb}, \varrho^{wb}, \varrho^{sa}, \varrho^{wa}\}$ values are defined in (5.13) as the strong and weak constraints before and after the variation analysis with $0 < \epsilon \ll 1$.

$$\varrho(h_l) = \max\{\varrho^{sb}, \varrho^{wb}, \varrho^{sa}, \varrho^{wa}\}. \quad (5.12)$$

$$\varrho^{sb} = \infty, \text{ if } h_l \text{ does not accomplish a strong constraint before variation analysis.} \quad (5.13a)$$

$$\varrho^{wb} = \epsilon, \text{ if } h_l \text{ does not accomplish a weak constraint before variation analysis.} \quad (5.13b)$$

$$\varrho^{sa} = -\epsilon, \text{ if } h_l \text{ does not accomplish a strong constraint after variation analysis.} \quad (5.13c)$$

$$\varrho^{wa} = -\infty, \text{ if } h_l \text{ does not accomplish a weak constraint after variation analysis.} \quad (5.13d)$$

In our experiments, the strong constraints are the saturation condition in all the transistors while the weak constraints are the goals in the circuit performances. The $\varrho(h_l)$ function always takes the maximum value among $\{\varrho^{sb}, \varrho^{wb}, \varrho^{sa}, \varrho^{wa}\}$ regardless the constraint that are not accomplished. Finally, the optimization is accomplished with all the constraints accomplished before and after the variation analysis ($h(\mathbf{x}) = 0$).

5.4.3 Application Example

There is performed the optimization of the FC OTA exposed before in Chapter 4, however this time by including a yield analysis. The encoding for the circuit is the same depicted in Table 4.1 and the optimization is performed with the same variables encoding, voltages, current bias and transistor technology. In our experiments we included the saturation condition in all transistors as constraints. $\mathbf{f}(\mathbf{x})$ is the vector formed by:

- $f_1(\mathbf{x}) = -1 * \text{Gain}$.
- $f_2(\mathbf{x}) = -1 * \text{GBW}$.
- $f_3(\mathbf{x}) = \text{Input referred noise}$.
- $f_4(\mathbf{x}) = \text{Input voltage offset}$.
- $f_5(\mathbf{x}) = \text{Settling time}$.
- $f_6(\mathbf{x}) = -1 * \text{Slew rate}$.
- $f_7(\mathbf{x}) = \text{Power consumption}$.

These experiments, besides to include the CBBA before the optimization process and the yield analysis for each solution that accomplishes the constraints, all the design values (length and width of the transistors) are discretized according to the technology (in this case $0.09\mu m$) with the aim to make easier the scaling through other technologies.

The optimization procedure is performed with a population size of 210 along 250 generations and it is compared the optimization when is used classical Monte Carlo (MC) and OCBA. For the classical MC is set 100 iterations, and for OCBA, the ψ_T values is 42,000 based on the worst case for MC and NSGA-II for one generation, that is when NSGA-II performs MC for all the solutions in its two populations (P and Q).

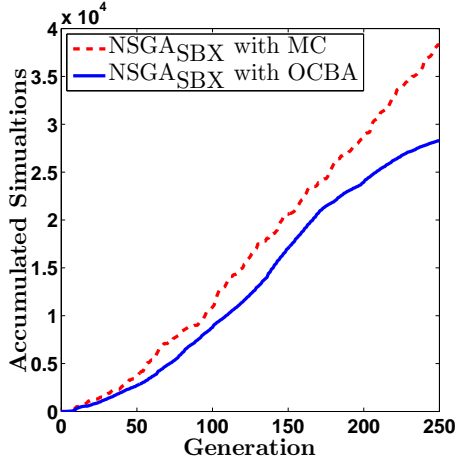
Figures 5.12- 5.14 show the simulations accumulated vs. generation performed by the yield analysis along all the optimization, for NSGA-II, MOEAD and MOPSO using MC and OCBA. Table 5.6 lists the results of the optimization by selecting the solutions with the highest yield for each experiment. The upper part of the table shows the objectives and below there are listed the values for the design variables. At the bottom of the table are listed the yield for each solution and the total number the simulations performed by the optimization. In all the cases, the yield is improved and the number of simulations are reduced when OCBA is used.

NSGA_{SBX} with OCBA improves the gain, GBW, ST and SR by increasing the yield 1.5x and reducing 25% the number of simulations compared with CM as shows Figure 5.12(a). NSGA_{DE} almost exhibits the same number of simulations (Figure 5.12(b)) with and without OCBA, however when OCBA is used the yield is improved 1.6x and the gain, GBW, ST and SR are also improved.

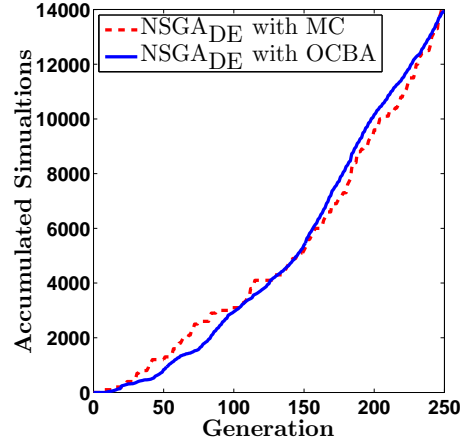
For MOEAD_{SBX}, OCBA shows a yield improvement of 1.8x and a reduction of simulations around 55% (Figure 5.13(a)) compared with MC, while the gain, GBW, Noise, ST and SR were improved. The case of MOEAD_{DE} is similar to MOEAD_{SBX} regarding to the high reduction of simulations, but this time OCBA achieves a 59 % of reduction of simulations (Figure 5.13(b)) and the yield is increased almost 1.2x by improving the GBW, ST and SR.

Finally, MOPSO with OCBA exhibits the highest reduction in the number of simulations and improves the gain, GBW, ST and SR. This time the reduction in the number of simulations along all the optimization process is almost 85% (Figure 5.14) and the yield improves more

than 1.5x when OCBA is used compared with MC.

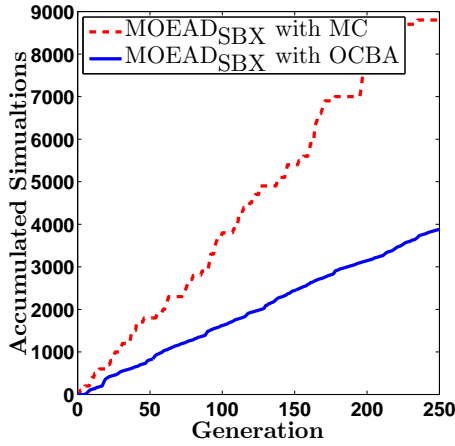


(a) NSGA-II_{SBX} Yield Aware Optimization.

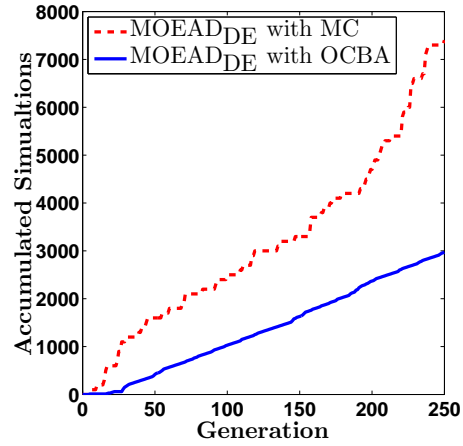


(b) NSGA-II_{DE} Yield Aware Optimization.

Figure 5.12: Accumulated simulations for the FC OTA with and without OCBA for NSGA-II.



(a) MOEAD_{SBX} Yield Aware Optimization.



(b) MOEAD_{DE} Yield Aware Optimization.

Figure 5.13: Accumulated simulations for the FC OTA with and without OCBA for MOEAD.

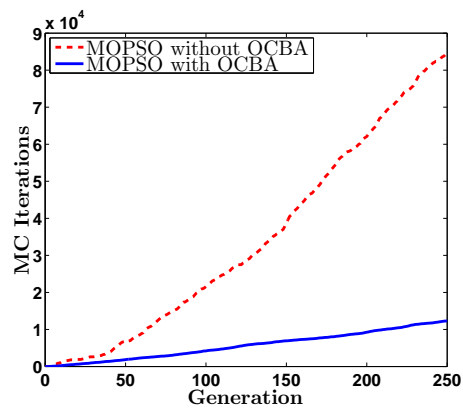


Figure 5.14: Accumulated simulations for the FC OTA with and without OCBA for MOPSO.

Table 5.6: Optimal variation-aware solutions for the FC OTA.

	NSGA _{SBX}		NSGA _{DE}		MOEAD _{SBX}		MOEAD _{DE}		MOPSO	
	MC	OCBA	MC	OCBA	MC	OCBA	MC	OCBA	MC	OCBA
Objectives										
Gain [dB]	45	49	56	50	45	47	53	49	45	51
GBW [MHz]	60	74	39	50	42	70	55	72	59	76
Noise [μ Vrms]	70	75	85	88	123	87	73	77	68	69
Offset [m V]	1.1	4.2	3.2	3.4	2.0	3.9	3.9	3.9	1.1	4.9
ST [ns]	29	16	38	25	35	17	24	17	31	14.8
SR [V/ μ s]	35	41	19	29	23	40	26	41	33	48
PW [mW]	3.8	4.2	3.2	3.8	4.2	4.4	3.6	4.2	3.8	4.0
Variables										
$L1[\mu m]$	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18	0.18
$L2[\mu m]$	0.18	0.18	0.81	0.45	0.63	0.18	0.45	0.18	0.18	0.18
$W1[\mu m]$	128.25	123.12	120.15	102.06	112.32	106.2	126.27	112.77	129.96	129.96
$W2[\mu m]$	44.01	47.43	44.28	43.29	79.2	46.98	44.46	43.29	43.29	43.29
$W3[\mu m]$	14.4	13.77	15.48	11.16	11.61	11.52	15.66	13.05	14.67	14.67
$W4[\mu m]$	82.53	86.4	75.51	72.63	86.49	83.97	66.33	78.93	86.58	75.78
$W5[\mu m]$	0.99	11.52	30.6	57.33	16.65	24.93	51.93	12.69	0.9	30.51
Total Yield Simulations	38300	28274	14000	13922	8800	3874	7400	2970	84300	12290
Yield	47 %	72 %	44 %	73 %	29 %	54 %	30 %	36 %	60 %	90 %

5.5 Summary

This Chapter was devoted to explore the variation aware optimization of analog circuits. The variation analysis is an important decision point because before this analysis, all the target specifications are accomplished but if we select those solutions without taking into account their variations, in the fabrication process there is a strong possibility that the designed circuits do not guarantee optimal performances. In this manner, it is important to be able to discriminate those solutions that are not really feasible.

The use of finite differences and Richardson extrapolation allows us to calculate the partial derivatives for multi-parameter sensitivity analysis, without an explicit mathematical expression. Our proposed approach based on the multi-parameter sensitivity was tested on the RFC OTA, and there were found the best solutions that accomplish all the target specifications for each objective and they improve the performances already published in [99] with the lowest multi-parameter sensitivity.

The shortcomings of this approach are related with the fact that the number of sensitivities increase dramatically with the number of design variables or circuit performances. Also this approach may be stalled when it is desired to find more accurate results, due to the significant digits that the simulator can handle.

Also, it has been performed a Non-Worst case approach based on MC simulations by introducing an optimal simulation budget allocation technique called OCBA. We showed that this approach allows always improving the yield and sometimes reducing the number of simulations required for a typical MC approach in the optimization of analog circuits with EAs.

The shortcoming of this approach is that it requires the transistor models from the factory and the complexity of these models and the number of simulations exhibit a strong dependence on the accuracy. To overcome the issue of the efficiency in this sort of methodology, the OCBA approach has shown an alternative reducing the number of simulations up to 85% and improving the yield up to 1.8x compared with a classical MC technique. For introducing the OCBA into the circuit optimization the only extra parameters are the maximum and initial number of simulations for the yield analysis.

Chapter 6

Conclusions

The analog circuit biasing and sizing can be considered as a non-linear problem with multiple objectives, dozens of variables and many constraints. The analog designers need to invest a lot of time to gain enough experience in a given circuit to accomplish all the design requirements. Furthermore, after a design is biased and sized properly, the process variations in the fabrication of that design might make that it can be rejected because its low tolerance to variations. A useful tool to cope with this complex problem is the use of EDA optimization tools. Those tools are devoted to help to the designers to gain insight into a design, find its limitations and ensure that the selected design is a feasible one because accomplish all the requirements.

Although several optimization tools exist both in the academic and industrial fields, that solve circuit biasing and sizing issues, they still have to overcome some design shortcomings. For instance, EDA tools have been developed for designing a specific set of circuits and also by optimizing a specific set of objectives. Some of them lost accuracy while improving their efficiency by using behavioral models. Besides, when compared with a circuit simulator their results may differ. Not all EDA tools are able to handle any transistor model and or technology, neither to include a process variations procedure to ensure feasible solutions. To cope with these circuit design challenges, this Thesis showed that EAs are a good choice to perform circuit optimization. Three EAs were described and applied, namely: NSGA-II, MOEAD and MOPSO. Along the chapters, several optimization examples were presented for different circuits to highlight the flexibility and suitability of EAs in varying the number of optimized

variables, constraints and objectives. Also, those experiments were performed with different transistor models and technologies, with any extra programming. All the results provided by EAs are trustworthy because they become directly from the circuit simulator, then all the accuracy depends on the simulator accuracy. By comparing the performance of each EA along all the examples, it is possible to notice that there is no algorithm superior to other.

However, the circuit optimization still requires a continuing search process to finding biased solutions, because it is not possible to optimize the circuit performances while not having biased solutions. This process depends strongly of the initial search space and may invest a lot of time. In this manner, this Thesis introduced a CBBA method where the basic idea is to choose the appropriate limits over the search space for the encoded variables in the sizing process that takes place in the analog circuit optimization. In this manner, it can be possible to achieve a successful optimization without having a deep insight in the circuit and without waste of time on searching values that are not feasible due to the wrong biasing of the circuit. To show the usefulness of the CBBA, there were presented circuit optimization examples performed with NSGA-II, MOEAD and MOPSO and finding that CBBA helps to accomplish the objective specifications compared when CBBA is not used and/or reduce the number of generations required to find an optimal solution.

Regarding to the process variation analysis into the circuit optimization, there were exposed two different approaches devoted to incorporate a Non-Worst case methodology taking into account the fabrication process variations, showing again the flexibility of the EAs to incorporate extra modules. The first method is based on the multi-parameter sensitivity of the circuit devices and the second one is based on MC simulations by using the OCBA approach to improve the yield and reducing the number of simulations required to achieve the optimization.

The use of EDA tools usually is highly impacted by three factors: the operating system, the simulator and the software interface. Those issues were taken into account to develop a software tool that gathers all the advantages of EAs, CBBA and OCBA to the optimization of analog circuits. In this Thesis a software tool was developed with PERL which is an open source language available for the most common operating systems. In the same direction, an evaluation module was programmed to perform the circuit simulations with any simulator having an

output file with the circuit measurements. Finally, the developed GUI facilitates its use for the analog circuit designers allowing entering all the optimization parameters and performing the optimization all on the same screen.

It is possible to summarize the next contributions:

- A multi-EA circuit optimizer platform.
- Highest-quality circuit performance tradeoffs.
- A new current-branches-bias assignment (CBBA) approach has been introduced in order to accelerate the sizing process of an analog integrated circuit composed of MOSFETs.
- Minimizing computational effort in the multi-objective analog circuit optimization by using OCBA.
- A portable software for analog circuit optimization based on EAs.

As a conclusion, we believe that the proposed approach is a powerful tool to enhance analog circuit design through generating feasible solutions accomplishing target specifications, with the possibility of improving the results and the efficiency by using the CBBA and OCBA methods.

Also, it is possible to choose among different variables encodings, to explore the best performances of an analog IC, and including support variations so that the optimal performance of a circuit design can be guaranteed.

Finally, as the main future work, it is possible to highlight the research on the local and global optimization for reducing the run time and the use of design of experiments such as Latin Hypercube or Quasi-Monte Carlo with the aim to reduce the number of simulations for the yield optimization.

Appendix A

Circuit Optimizer Software Tool

This section is devoted to show the circuit optimizer software tool developed to perform all the circuit optimizations in this Thesis. The software tool is composed by the code of NSGA-II, MOEAD and MOPSO adapted for the circuit optimization and able to choose the use of CBBA and/or yield analysis (performed with classical Monte Carlo or OCBA). Figure A.1 shows the main modules on the left side and on the right side shows the inputs to each module. The general parameters are the population size and the maximum number of generations. The CBBA module needs the circuit netlist and this module can be optional. The initialization module is devoted to generate the first population where each variable is delimited by a minimum and maximum value (in the case that CBBA is executed, the bounds of some variables are modified according to the different current branches). The optimization module selects an EA (NSGA-II, MOEAD or MOPSO) and a recombination operator (SBX or DE), according with these choices are set different parameters. The sub-module of circuit simulation requires the circuit netlist, the objective functions, the constraints and the circuit simulator. Finally the sub-module of yield analysis is optional but requires the circuit netlist adapted to Monte Carlo simulations, choosing between classical Monte Carlo or OCBA analysis and a maximum number of iterations for the yield analysis.

In this manner the software developed, links all the modules with its input parameters making easier to perform a circuit optimization without programming and by feeding all the parameters. The graphical interface was programmed with TCL and all the optimization is pro-

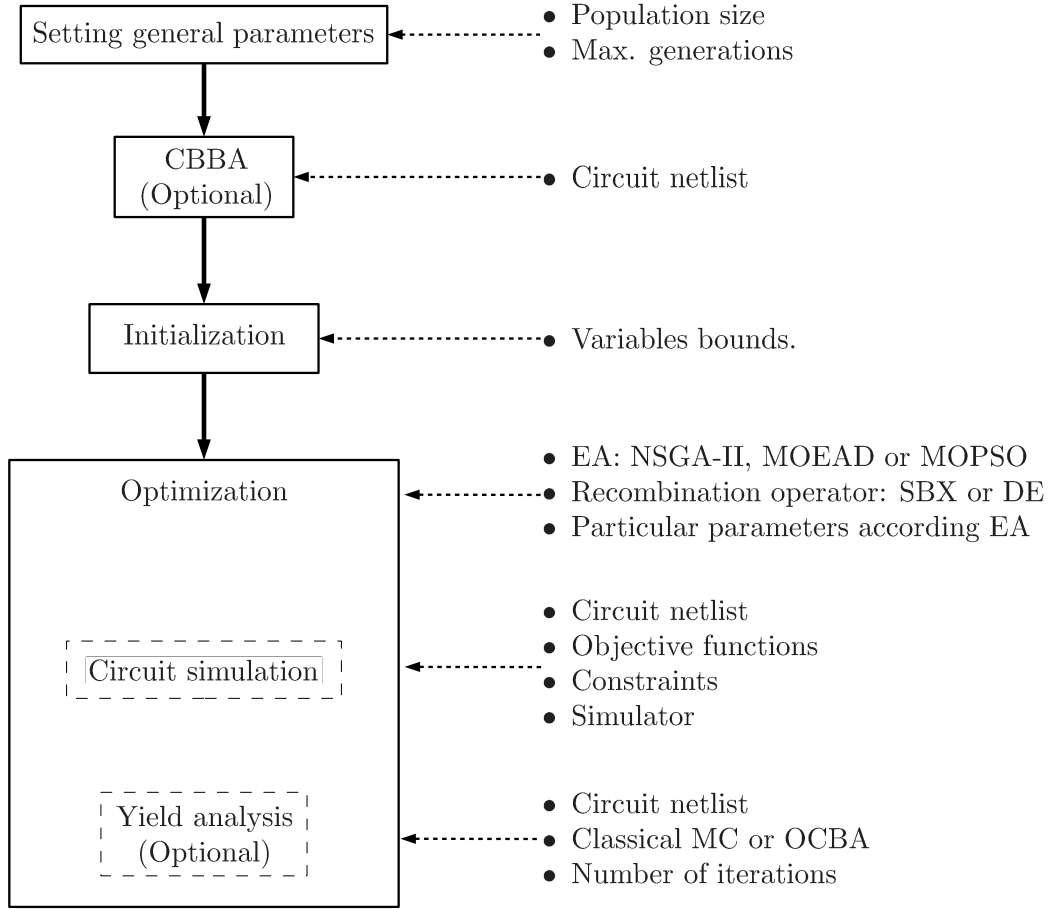


Figure A.1: Software tool modules and inputs.

grammed in PERL, these features allow to run this software on any operative system, also it is used HSPICE[®] as circuit simulator, and any version for this simulator can be used.

A.1 Software input sections

The software consist of a main window that contains all the inputs parameters for each module. The main window is depicted on Figure A.2 and is devoted to ask to the user all the parameters along its sections:

- HSPICE.
- Variables.

- Objectives.
- Constraints.
- Optimization Parameters.
- OCBA and Yield.

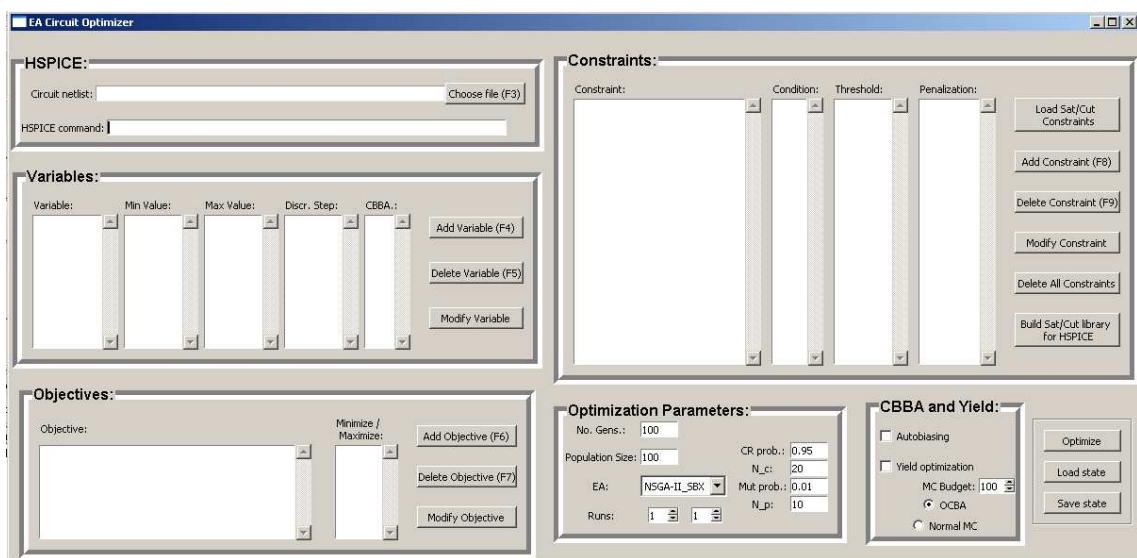


Figure A.2: Software tool main window.

A.1.1 HSPICE Section

The HSPICE section has two inputs:

- Circuit netlist: it admits any extension (usually .sp or .cir) for the file and it has the circuit netlist to simulate it with HSPICE® along the optimization.
- HSPICE command: for WINDOWS® systems it is specified the full path of the simulator and for the other operative systems only is specified the command i.e.: *hspice* >.

Figure A.3: HSPICE section.

A.1.2 Variables Section

The variables section allows adding, modifying or deleting variables. It has five inputs as shows Figure A.4 and the order how appear the variables here is the same order that is going to be displayed for the simulator:

- Variable: variable name, it can be any combination of characters and it is used to identify the variable.
- Min Value: the minimum limit value for the variable for both, the initialization and the optimization.
- Max Value: the maximum limit value for the variable for both, the initialization and the optimization.
- Discr. Step: if it is desired that the variables accomplish discrete values, it accepts any value and for continuos variables can be set a very small value as $1E - 10$.
- CBBA: if the variable is going to be partitioned with the CBBA procedure.

To add a new variable there are required the five inputs as shows Figure A.5 where the variable name is $W1$ (it represents the width of some transistors in the netlist), and $0.9\mu m \leq W1 \leq 130\mu m$ with $0.09\mu m$ as discrete step, and it is desired that this variable has partitioned by the CBBA procedure.

Figure A.4: Variables section.

Figure A.5: Add new variable.

A.1.3 Objectives Section

The objectives section allows adding, modifying or deleting objectives (they can be entered in any order) and it has two main inputs as shows Figure A.6 :

- Objective: objective name, it can be any combination of characters and it is used to identify the objective within the simulation output listing.
- Minimize/Maximize: it indicates if the objective is desired maximized or minimized for the optimization.

In addition to these two inputs, there exists other three inputs if it is desired to set a constraint to the objectives as shows Figure A.6. Those inputs are exposed on the constraints section.



Figure A.6: Objectives section.

To add a new variable there are required the two main inputs as shows Figure A.7 where the objective name is *gain* (it represents the gain of the circuit measured by the simulator and it lists the value in the output listing), it is desired to maximize its value setting as constraint that must be great than 50dB (the units are the same that the simulator are measuring) and with a penalization of 1 (the penalization is explained on constraints section).

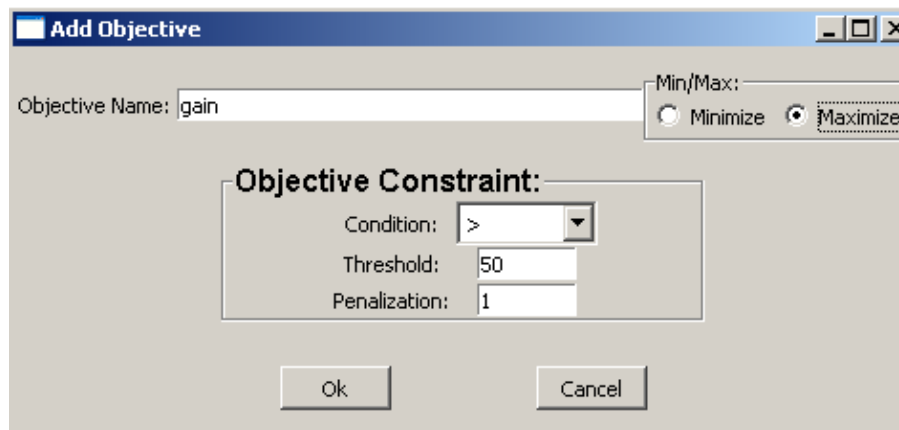


Figure A.7: Add new objective.

A.1.4 Constraint Section

The constraints section allows adding, modifying or deleting constraints (they can be entered in any order) and it has four inputs as shows Figure A.8 :

- Constraint: constraint name, it can be any combination of characters and it is used to identify the constraint within the simulation output listing.

- Condition: it indicates if the constraints must be less or greater than a threshold value.
- Threshold: it is the threshold value for the constraint condition.
- Penalization: it is the value which penalizes when a constraint is not accomplished. In general all the constraints can have the same penalization value, however it is possible to handling different values with the aim to identify among different constraint hierarchy levels, for instance, the saturation of the transistors can have a great value (i.e. 100) because has a higher hierarchy level than others as objective constraint that have less values (i.e. 1).

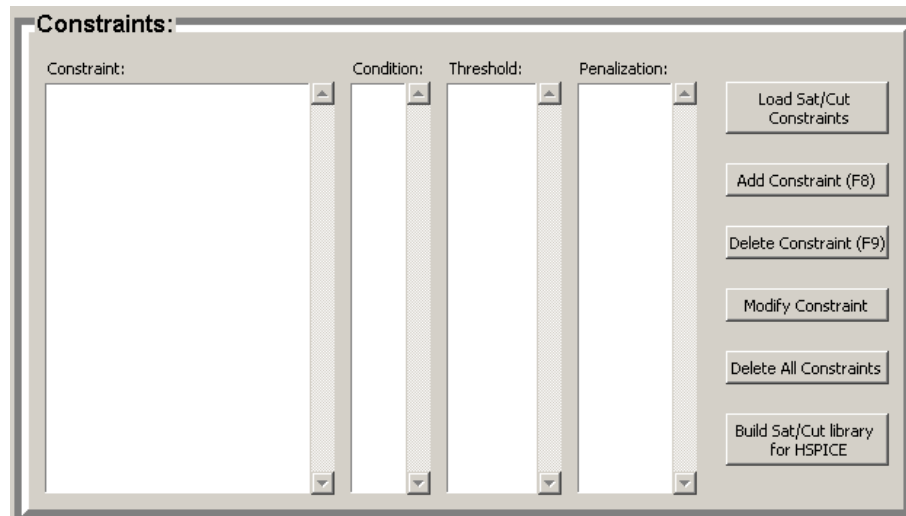


Figure A.8: Constraints section.

To add a new constraint there are required the four inputs as shows Figure A.9 where the objective name is *gain* (it represents the gain of the circuit measured by the simulator and it lists the value in the output listing), it must be greater than 50dB (the units are the same that the simulator are measuring) and with a penalization of 1.

There exist other two options in this section related to the saturation and lineal condition of the transistors constraints:

- Load Sat/Cut Constraints: with this option, all the saturation and lineal of transistor constraints are entered in the list of constraints from the circuit netlist. This option automat-

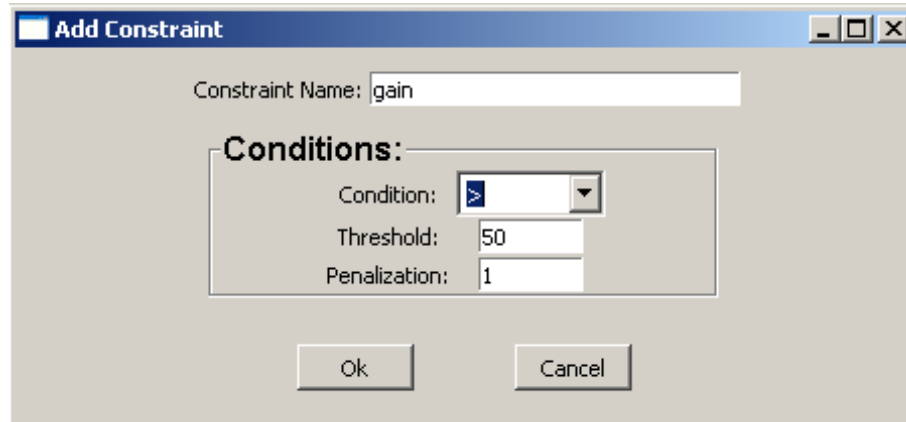


Figure A.9: Add new constraint.

ically detects all the transistor names and enters the list of constraints for saturation and lineal condition for each transistor as shows Figure A.10.

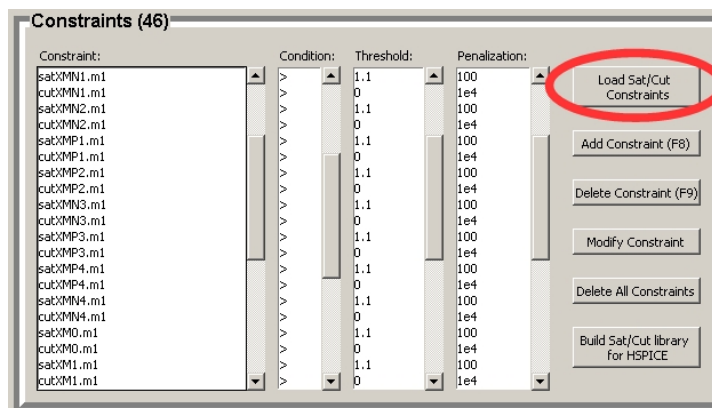
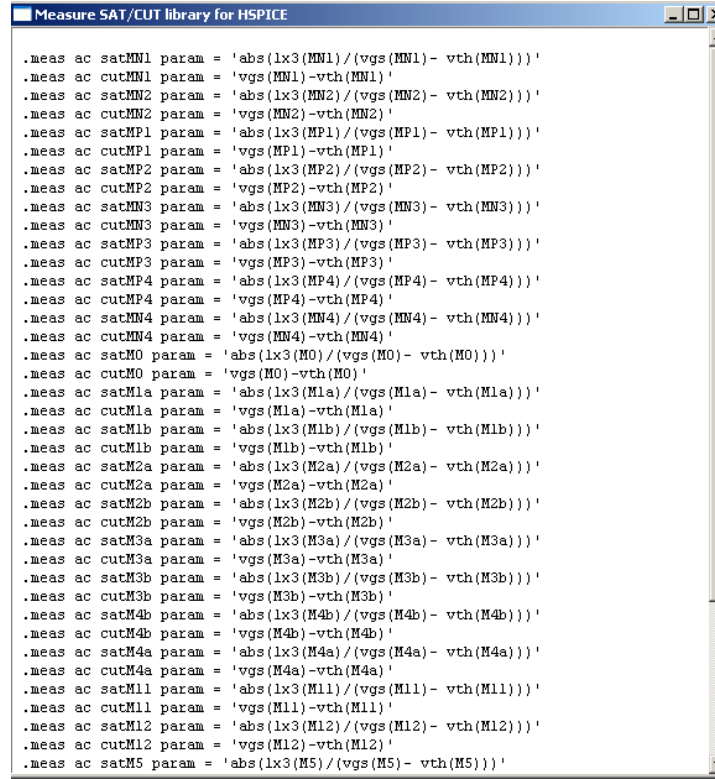


Figure A.10: Load Sat/Cut Constraints.

- **Build Sat/Cut library for HSPICE:** this option builds automatically a list with the instructions for measuring in HSPICE the saturation and lineal condition of all the transistors in the circuit netlist, as shows Figure A.11.

A.1.5 Optimization parameters section

This section is devoted to set all the general and particular parameters related with the optimization procedure and the EA chosen.



```

.meas ac satMN1 param = 'abs(1x3(MN1))/(vgs(MN1)- vth(MN1))'
.meas ac cutMN1 param = 'vgs(MN1)-vth(MN1)'
.meas ac satMN2 param = 'abs(1x3(MN2))/(vgs(MN2)- vth(MN2))'
.meas ac cutMN2 param = 'vgs(MN2)-vth(MN2)'
.meas ac satMP1 param = 'abs(1x3(MP1))/(vgs(MP1)- vth(MP1))'
.meas ac cutMP1 param = 'vgs(MP1)-vth(MP1)'
.meas ac satMP2 param = 'abs(1x3(MP2))/(vgs(MP2)- vth(MP2))'
.meas ac cutMP2 param = 'vgs(MP2)-vth(MP2)'
.meas ac satMN3 param = 'abs(1x3(MN3))/(vgs(MN3)- vth(MN3))'
.meas ac cutMN3 param = 'vgs(MN3)-vth(MN3)'
.meas ac satMP3 param = 'abs(1x3(MP3))/(vgs(MP3)- vth(MP3))'
.meas ac cutMP3 param = 'vgs(MP3)-vth(MP3)'
.meas ac satMP4 param = 'abs(1x3(MP4))/(vgs(MP4)- vth(MP4))'
.meas ac cutMP4 param = 'vgs(MP4)-vth(MP4)'
.meas ac satMN4 param = 'abs(1x3(MN4))/(vgs(MN4)- vth(MN4))'
.meas ac cutMN4 param = 'vgs(MN4)-vth(MN4)'
.meas ac satM0 param = 'abs(1x3(M0))/(vgs(M0)- vth(M0))'
.meas ac cutM0 param = 'vgs(M0)-vth(M0)'
.meas ac satM1a param = 'abs(1x3(M1a))/(vgs(M1a)- vth(M1a))'
.meas ac cutM1a param = 'vgs(M1a)-vth(M1a)'
.meas ac satM1b param = 'abs(1x3(M1b))/(vgs(M1b)- vth(M1b))'
.meas ac cutM1b param = 'vgs(M1b)-vth(M1b)'
.meas ac satM2a param = 'abs(1x3(M2a))/(vgs(M2a)- vth(M2a))'
.meas ac cutM2a param = 'vgs(M2a)-vth(M2a)'
.meas ac satM2b param = 'abs(1x3(M2b))/(vgs(M2b)- vth(M2b))'
.meas ac cutM2b param = 'vgs(M2b)-vth(M2b)'
.meas ac satM3a param = 'abs(1x3(M3a))/(vgs(M3a)- vth(M3a))'
.meas ac cutM3a param = 'vgs(M3a)-vth(M3a)'
.meas ac satM3b param = 'abs(1x3(M3b))/(vgs(M3b)- vth(M3b))'
.meas ac cutM3b param = 'vgs(M3b)-vth(M3b)'
.meas ac satM4b param = 'abs(1x3(M4b))/(vgs(M4b)- vth(M4b))'
.meas ac cutM4b param = 'vgs(M4b)-vth(M4b)'
.meas ac satM4a param = 'abs(1x3(M4a))/(vgs(M4a)- vth(M4a))'
.meas ac cutM4a param = 'vgs(M4a)-vth(M4a)'
.meas ac satM11 param = 'abs(1x3(M11))/(vgs(M11)- vth(M11))'
.meas ac cutM11 param = 'vgs(M11)-vth(M11)'
.meas ac satM12 param = 'abs(1x3(M12))/(vgs(M12)- vth(M12))'
.meas ac cutM12 param = 'vgs(M12)-vth(M12)'
.meas ac satM5 param = 'abs(1x3(M5))/(vgs(M5)- vth(M5))'

```

Figure A.11: Build Sat/Cut library for HSPICE.

The general parameters are:

- No. Gens.: it denotes the maximum number of generations in the optimization process.
- Population size: it fixes the number of individuals or solutions in each generation.
- Runs: it is possible to perform more than one run of the optimization to evaluate the results over several runs.

The particular parameters depends on the EA chosen in the combobox as depicts Figure A.12, and also are depicted the different particular parameters according the EA selected.

A.1.6 CBBA and Yield section

This section has the option to include CBBA and yield analysis by selecting the checkboxes. For the yield optimization option, it is required to set if is desired a normal Monte Carlo analysis

Optimization Parameters:

No. Gens.: 100

Population Size: 100

EA: NSGA-II_SBX

Runs: 1 1

CR prob.: 0.95

N_c: 20

Mut prob.: 0.01

N_p: 10

(a) NSGA-II with SBX as recombination operator.

Optimization Parameters:

No. Gens.: 100

Population Size: 100

EA: NSGA-II_DE

Runs: 1 1

CR prob.: 0.95

Fx: 0.4

Mut prob.: 0.01

(b) NSGA-II with DE as recombination operator.

Optimization Parameters:

No. Gens.: 100

H: 10 Calc N

Population Size: 100

EA: MOEAD_SBX

Runs: 1 1

CR prob.: 0.95

N_c: 20

Mut prob.: 0.01

N_p: 10

T: 20

(c) MOEAD with SBX as recombination operator.

Optimization Parameters:

No. Gens.: 100

H: 10 Calc N

Population Size: 100

EA: MOEAD_DE

Runs: 1 1

CR prob.: 0.95

Fx: 0.4

Mut prob.: 0.01

T: 20

(d) MOEAD with DE as recombination operator.

Optimization Parameters:

No. Gens.: 100

Population Size: 100

EA: PSO

Runs: 1 1

kw: 0.4

(e) MOPSO .

Figure A.12: General and particular parameters for each EA.

or OCBA for yield analysis, and finally setting the budget simulations as depicts Figure A.13.

CBBA and Yield:

☐ Autobiassing

☐ Yield optimization

MC Budget: 100

☒ OCBA

☐ Normal MC

Figure A.13: Build Sat/Cut library for HSPICE.

A.2 Buttons section

This is the last section in the software tool and has three buttons:

- **Optimize:** this button starts the optimization with all the inputs in all the sections by displaying the current generation and the best penalization in the current optimization to get an idea about the current optimization, the optimization objective is to find the less penalization (it is desired zero penalization).
- **Load State:** this button allows to load previous inputs in all the sections .
- **Save State:** this button allows to save into a file the current inputs of all the sections.

A.3 Software output

The final result of a optimization are a set of files, all of them called as the circuit netlist file as prefix (File_Name) and the number of run as suffix (the number 1 represents the first run, the number 2 represents the second and so on), which contains the results of the optimization (values, function objectives, penalization, yields). Next it is exposed the contents of each file for the first run:

- **File_Name_R1.rw:** it has all the historic of all individuals over all the generations.
- **File_Name_R1_aux.rw:** it has all the historic of all individuals of the last generation as backup to continue the optimization in case of necessary.
- **File_Name_REP1.rw:** it lists the best solutions for each generation.
- **File_Name_OUT.rw:** it lists all the best solutions of the last generation.

Appendix B

Transistor Models

B.1 Transistor Model for 0.35 μm technology

```
.MODEL MODN NMOS LEVEL=49
* -----
***** SIMULATION PARAMETERS *****
* -----
* format      : HSPICE
* model       : MOS BSIM3v3
* process     : C35
* revision    : 2;
* extracted   : B10866 ; 2002-12; ese(487)
* doc#        : ENG-182 REV_2
* -----
*                                     TYPICAL MEAN CONDITION
* -----
*
*          *** Flags ***
+MOBMOD =1.000e+00 CAPMOD =2.000e+00
+NOIMOD =3.000e+00
+VERSION=3.11
*          *** Threshold voltage related model parameters ***
+K1      =5.0296e-01
+K2      =3.3985e-02 K3      =-1.136e+00 K3B      =-4.399e-01
+NCH     =2.611e+17 VTH0     =4.979e-01
+VOFF    =-8.925e-02 DVT0    =5.000e+01 DVT1      =1.039e+00
+DVT2    =-8.375e-03 KETA    =2.032e-02
+PSCBE1  =3.518e+08 PSCBE2  =7.491e-05
+DVT0W   =1.089e-01 DVT1W   =6.671e+04 DVT2W     =-1.352e-02
```

```

*      *** Mobility related model parameters ***
+UA      =4.705e-12 UB      =2.137e-18 UC      =1.000e-20
+U0      =4.758e+02
*      *** Subthreshold related parameters ***
+DSUB    =5.000e-01 ETA0    =1.415e-02 ETAB    =-1.221e-01
+NFACTOR=4.136e-01
*      *** Saturation related parameters ***
+EM      =4.100e+07 PCLM    =6.948e-01
+PDIBLC1=3.571e-01 PDIBLC2=2.065e-03 DROUT    =5.000e-01
+A0      =2.541e+00 A1      =0.000e+00 A2      =1.000e+00
+PVAG    =0.000e+00 VSAT    =1.338e+05 AGS      =2.408e-01
+B0      =4.301e-09 B1      =0.000e+00 DELTA    =1.442e-02
+PDIBLCB=3.222e-01
*      *** Geometry modulation related parameters ***
+W0      =2.673e-07 DLC      =3.0000e-08
+DWC      =9.403e-08 DWB      =0.000e+00 DWG      =0.000e+00
+LL      =0.000e+00 LW      =0.000e+00 LWL      =0.000e+00
+LLN      =1.000e+00 LWN      =1.000e+00 WL      =0.000e+00
+WW      =-1.297e-14 WWL      =-9.411e-21 WLN      =1.000e+00
+WWN      =1.000e+00
*      *** Temperature effect parameters ***
+TNOM    =27.0 AT      =3.300e+04 UTE      =-1.800e+00
+KT1      =-3.302e-01 KT2      =2.200e-02 KT1L      =0.000e+00
+UA1      =0.000e+00 UB1      =0.000e+00 UC1      =0.000e+00
+PRT      =0.000e+00
*      *** Overlap capacitance related and dynamic model parameters ***
+CGDO    =1.300e-10 CGSO    =1.200e-10 CGBO    =1.100e-10
+CGDL    =1.310e-10 CGSL    =1.310e-10 CKAPPA =6.000e-01
+CF      =0.000e+00 ELM      =5.000e+00
+XPART    =1.000e+00 CLC      =1.000e-15 CLE      =6.000e-01
*      *** Parasitic resistance and capacitance related model parameters ***
+RDSW    =3.449e+02
+CDSC    =0.000e+00 CDSCB    =1.500e-03 CDSCD    =1.000e-03
+PRWB    =-2.416e-01 PRWG    =0.000e+00 CIT      =4.441e-04
*      *** Process and parameters extraction related model parameters ***
+TOX      =7.575e-09 NGATE    =0.000e+00
+NLX      =1.888e-07
+XL      =0.000e+00 XW      =0.000e+00
*      *** Substrate current related model parameters ***
+ALPHA0   =0.000e+00 BETA0    =3.000e+01
*      *** Noise effect related model parameters ***
+AF      =1.3600e+00 KF      =5.1e-27 EF      =1.000e+00

```

```

+NOIA    =1.73e+19 NOIB    =7.000e+04 NOIC    =-5.64e-13
*        *** Common extrinsic model parameters ***
+ACM      =2
+RD       =0.000e+00 RS      =0.000e+00 RSH      =7.000e+01
+RDC      =0.000e+00 RSC      =0.000e+00
+LINT     =-5.005e-08 WINT    =9.403e-08
+LDIF     =0.000e+00 HDIF     =8.000e-07 WMLT      =1.000e+00
+LMLT     =1.000e+00 XJ       =3.000e-07
+JS       =1.000e-05 JSW      =0.000e+00 IS        =0.000e+00
+N        =1.000e+00 NDS      =1000.
+VNDS     =-1.000e+00 CBD=0.000e+00 CBS=0.000e+00 CJ=9.400e-04 CJSW=2.500e-10
+FC       =0.000e+00 MJ       =3.400e-01 MJSW      =2.300e-01 TT        =0.000e+00
+PB       =6.900e-01 PHP      =6.900e-01
* -----

```

```

.MODEL MODP PMOS LEVEL=49

```

```

* -----
***** SIMULATION PARAMETERS *****
* -----
* format      : HSPICE
* model       : MOS BSIM3v3
* process     : C35
* revision    : 2;
* extracted   : C64685 ; 2002-12; ese(487)
* doc#        : ENG-182 REV_2
* -----
*
*                                     TYPICAL MEAN CONDITION
* -----
*
*        *** Flags ***
+MOBMOD =1.000e+00 CAPMOD =2.000e+00
+NOIMOD =3.000e+00
+VERSION=3.11
*        *** Threshold voltage related model parameters ***
+K1      =5.9959e-01
+K2      =-6.038e-02 K3      =1.103e+01 K3B      =-7.580e-01
+NCH     =9.240e+16 VTH0     =-6.915e-01
+VOFF    =-1.170e-01 DVT0    =1.650e+00 DVT1      =3.868e-01
+DVT2    =1.659e-02 KETA     =-1.440e-02
+PSCBE1  =5.000e+09 PSCBE2  =1.000e-04

```

```

+DVT0W =1.879e-01 DVT1W =7.335e+04 DVT2W =-6.312e-03
*      *** Mobility related model parameters ***
+UA     =5.394e-10 UB      =1.053e-18 UC      =1.000e-20
+U0     =1.482e+02
*      *** Subthreshold related parameters ***
+DSUB   =5.000e-01 ETA0    =2.480e-01 ETAB    =-3.917e-03
+NFACTOR=1.214e+00
*      *** Saturation related parameters ***
+EM     =4.100e+07 PCLM    =3.184e+00
+PDIBLC1=1.000e-04 PDIBLC2=1.000e-20 DROUT   =5.000e-01
+A0     =5.850e-01 A1      =0.000e+00 A2      =1.000e+00
+PVAG   =0.000e+00 VSAT    =1.158e+05 AGS     =2.468e-01
+B0     =8.832e-08 B1      =0.000e+00 DELTA   =1.000e-02
+PDIBLCB=1.000e+00
*      *** Geometry modulation related parameters ***
+W0     =1.000e-10 DLC     =2.4500e-08
+DWC    =3.449e-08 DWB     =0.000e+00 DWG     =0.000e+00
+LL     =0.000e+00 LW      =0.000e+00 LWL     =0.000e+00
+LLN    =1.000e+00 LWN     =1.000e+00 WL      =0.000e+00
+WW     =1.894e-16 WWL     =-1.981e-21 WLN     =1.000e+00
+WWN    =1.040e+00
*      *** Temperature effect parameters ***
+TNOM   =27.0 AT          =3.300e+04 UTE      =-1.300e+00
+KT1    =-5.403e-01 KT2   =2.200e-02 KT1L    =0.000e+00
+UA1    =0.000e+00 UB1    =0.000e+00 UC1     =0.000e+00
+PRT    =0.000e+00
*      *** Overlap capacitance related and dynamic model parameters ***
+CGDO   =8.600e-11 CGSO    =8.600e-11 CGBO    =1.100e-10
+CGDL   =1.080e-10 CGSL    =1.080e-10 CKAPPA  =6.000e-01
+CF     =0.000e+00 ELM     =5.000e+00
+XPART  =1.000e+00 CLC     =1.000e-15 CLE     =6.000e-01
*      *** Parasitic resistance and capacitance related model parameters ***
+RDSW   =1.033e+03
+CDSC   =2.589e-03 CDSCB   =2.943e-04 CDSCD   =4.370e-04
+PRWB   =-9.731e-02 PRWG   =1.477e-01 CIT     =0.000e+00
*      *** Process and parameters extraction related model parameters ***
+TOX    =7.754e-09 NGATE   =0.000e+00
+NLX    =1.770e-07
+XL     =0.000e+00 XW      =0.000e+00
*      *** Substrate current related model parameters ***
+ALPHA0 =0.000e+00 BETA0   =3.000e+01
*      *** Noise effect related model parameters ***

```

```

+AF      =1.48e+00 KF      =8.5e-27 EF      =1.000e+00
+NOIA    =1.52e+18 NOIB    =7.75e+03 NOIC    =5.0e-13
*        *** Common extrinsic model parameters ***
+ACM      =2
+RD       =0.000e+00 RS      =0.000e+00 RSH      =1.290e+02
+RDC      =0.000e+00 RSC      =0.000e+00
+LINT     =-7.130e-08 WINT    =3.449e-08
+LDIF     =0.000e+00 HDIF     =8.000e-07 WMLT      =1.000e+00
+LMLT     =1.000e+00 XJ      =3.000e-07
+JS       =9.000e-05 JSW      =0.000e+00 IS       =0.000e+00
+N        =1.000e+00 NDS      =1000.
+VNDS     =-1.000e+00 CBD=0.000e+00 CBS=0.000e+00 CJ=1.360e-03 CJSW=3.200e-10
+FC       =0.000e+00 MJ       =5.600e-01 MJSW      =4.300e-01 TT       =0.000e+00
+PB       =1.020e+00 PHP      =1.020e+00
* -----

```

B.2 Transistor Model for 180 nm technology

```

* DATE: May 21/09
* LOT: T92Y          WAF: 9103
* Temperature_parameters=Default
.MODEL MODN NMOS (
+VERSION = 3.1          TNOM   = 27          TOX      = 4.1E-9
+XJ       = 1E-7        NCH    = 2.3549E17    VTH0     = 0.3694303
+K1       = 0.5789116   K2     = 1.110723E-3   K3       = 1E-3
+K3B      = 0.0297124   W0     = 1E-7        NLX      = 2.037748E-7
+DVT0W    = 0          DVT1W   = 0          DVT2W    = 0
+DVT0     = 1.2953626   DVT1   = 0.3421545    DVT2     = 0.0395588
+U0       = 293.1687573 UA     = -1.21942E-9  UB       = 2.325738E-18
+UC       = 7.061289E-11 VSAT  = 1.676164E5    A0       = 2
+AGS      = 0.4764546   B0     = 1.617101E-7    B1       = 5E-6
+KETA     = -0.0138552  A1     = 1.09168E-3    A2       = 0.3303025
+RDSW     = 105.6133217 PRWG  = 0.5          PRWB     = -0.2
+WR       = 1          WINT    = 2.885735E-9   LINT     = 1.715622E-8
+XL       = 0          XW     = -1E-8        DWG      = 2.754317E-9
+DWB      = -3.690793E-9 VOFF  = -0.0948017    NFACTOR  = 2.1860065
+CIT      = 0          CDSC   = 2.4E-4        CDSCD    = 0
+CDSCB    = 0          ETA0   = 2.665034E-3    ETAB     = 6.028975E-5
+DSUB     = 0.0442223  PCLM  = 1.746064    PDIBLC1  = 0.3258185

```


+PDIBLC2 = 2.701992E-3	PDIBLCB = -0.1	DROUT = 0.9787232
+PSCBE1 = 4.494778E10	PSCBE2 = 3.672074E-8	PVAG = 0.0122755
+DELTA = 0.01	RSH = 7	MOBMOD = 1
+PRT = 0	UTE = -1.5	KT1 = -0.11
+KT1L = 0	KT2 = 0.022	UA1 = 4.31E-9
+UB1 = -7.61E-18	UC1 = -5.6E-11	AT = 3.3E4
+WL = 0	WLN = 1	WW = 0
+WWN = 1	WWL = 0	LL = 0
+LLN = 1	LW = 0	LWN = 1
+LWL = 0	CAPMOD = 2	XPART = 0.5
+CGDO = 8.58E-10	CGSO = 8.58E-10	CGBO = 1E-12
+CJ = 9.471097E-4	PB = 0.8	MJ = 0.3726161
+CJSW = 1.905901E-10	PBSW = 0.8	MJSW = 0.1369758
+CJSWG = 3.3E-10	PBSWG = 0.8	MJSWG = 0.1369758
+CF = 0	PVTH0 = -5.105777E-3	PRDSW = -1.1011726
+PK2 = 2.247806E-3	WKETA = -5.071892E-3	LKETA = 5.324922E-4
+PU0 = -4.0206081	PUA = -4.48232E-11	PUB = 5.018589E-24
+PVSAT = 2E3	PETA0 = 1E-4	PKETA = -2.090695E-3)

*

.MODEL MODP PMOS (LEVEL = 49
+VERSION = 3.1	TNOM = 27	TOX = 4.1E-9
+XJ = 1E-7	NCH = 4.1589E17	VTH0 = -0.3823437
+K1 = 0.5722049	K2 = 0.0219717	K3 = 0.1576753
+K3B = 4.2763642	W0 = 1E-6	NLX = 1.104212E-7
+DVT0W = 0	DVT1W = 0	DVT2W = 0
+DVT0 = 0.6234839	DVT1 = 0.2479255	DVT2 = 0.1
+U0 = 109.4682454	UA = 1.31646E-9	UB = 1E-21
+UC = -1E-10	VSAT = 1.054892E5	A0 = 1.5796859
+AGS = 0.3115024	B0 = 4.729297E-7	B1 = 1.446715E-6
+KETA = 0.0298609	A1 = 0.3886886	A2 = 0.4010376
+RDSW = 199.1594405	PRWG = 0.5	PRWB = -0.4947034
+WR = 1	WINT = 0	LINT = 2.93948E-8
+XL = 0	XW = -1E-8	DWG = -1.998034E-8
+DWB = -2.481453E-9	VOFF = -0.0935653	NFACTOR = 2
+CIT = 0	CDSC = 2.4E-4	CDSCD = 0
+CDSCB = 0	ETA0 = 3.515392E-4	ETAB = -4.804338E-4
+DSUB = 1.215087E-5	PCLM = 0.96422	PDIBLC1 = 3.026627E-3
+PDIBLC2 = -1E-5	PDIBLCB = -1E-3	DROUT = 1.117016E-4
+PSCBE1 = 7.999986E10	PSCBE2 = 8.271897E-10	PVAG = 0.0190118
+DELTA = 0.01	RSH = 8.1	MOBMOD = 1

+PRT	= 0	UTE	= -1.5	KT1	= -0.11
+KT1L	= 0	KT2	= 0.022	UA1	= 4.31E-9
+UB1	= -7.61E-18	UC1	= -5.6E-11	AT	= 3.3E4
+WL	= 0	WLN	= 1	WW	= 0
+WWN	= 1	WWL	= 0	LL	= 0
+LLN	= 1	LW	= 0	LWN	= 1
+LWL	= 0	CAPMOD	= 2	XPART	= 0.5
+CGDO	= 7.82E-10	CGSO	= 7.82E-10	CGBO	= 1E-12
+CJ	= 1.214428E-3	PB	= 0.8461606	MJ	= 0.4192076
+CJSW	= 2.165642E-10	PBSW	= 0.8	MJSW	= 0.3202874
+CJSWG	= 4.22E-10	PBSWG	= 0.8	MJSWG	= 0.3202874
+CF	= 0	PVTH0	= 5.167913E-4	PRDSW	= 9.5068821
+PK2	= 1.095907E-3	WKETA	= 0.0133232	LKETA	= -3.648003E-3
+PU0	= -1.0674346	PUA	= -4.30826E-11	PUB	= 1E-21
+PVSAT	= 50	PETA0	= 1E-4	PKETA	= -1.822724E-3

)

*

B.3 Transistor Model for 90 nm technology

```
.PARAM MC_SPHVT10_VTH0_MA_N = AGAUSS(0, 1, 1)
.PARAM MC_SPHVT10_U0_MA_N = AGAUSS(0, 1, 1)
.PARAM MC_SPHVT10_VTH0_MA_P = AGAUSS(0, 1, 1)
.PARAM MC_SPHVT10_U0_MA_P = AGAUSS(0, 1, 1)

.PARAM MC_SPHVT10_TOX_NP = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_CJS_N = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_CJSWS_N = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_CJSWGS_N = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_XW_N = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_XL_N = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_VTH0_N = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_K3_N = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_WLPE0_N = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_LPE0_N = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_RDSW_N = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_U0_N = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_WWL_N = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_CF_N = AGAUSS(0, sigma, 3)

*

.PARAM DTOXE_N_10_SPHVT = '2.2500E-09 * (0.0444 / 3 * MC_SPHVT10_TOX_NP) * PROCESS'
```

```

.PARAM DTOXP_N_10_SPHVT = '1.8680E-09 * (0.0535 / 3 * MC_SPHVT10_TOX_NP) * PROCESS'
.PARAM DXW_N_10_SPHVT = ' (3.00000E-09 / 3) * MC_SPHVT10_XW_N * PROCESS'
.PARAM DXL_N_10_SPHVT = ' (5.00000E-09 / 3) * MC_SPHVT10_XL_N * PROCESS'
.PARAM DVTH0_N_10_SPHVT = ' (1.50000E-02 / 3) * MC_SPHVT10_VTH0_N * PROCESS'
.PARAM DK3_N_10_SPHVT = ' (1.77000E+00 / 3) * MC_SPHVT10_K3_N * PROCESS'
.PARAM DWLPE0_N_10_SPHVT = ' (3.60000E-09 / 3) * MC_SPHVT10_WLPE0_N * PROCESS'
.PARAM DLPE0_N_10_SPHVT = ' (5.40000E-09 / 3) * MC_SPHVT10_LPE0_N * PROCESS'
.PARAM DCGDL_N_10_SPHVT='1.8420E-10*(1/(1+0.0444/3*MC_SPHVT10_TOX_NP)-1)*PROCESS'
.PARAM DCGSL_N_10_SPHVT='1.8420E-10*(1/(1+0.0444/3*MC_SPHVT10_TOX_NP)-1)*PROCESS'
.PARAM DCJS_N_10_SPHVT = '1.1830E-03 * (0.1 / 3 * MC_SPHVT10_CJS_N) * PROCESS'
.PARAM DCJSWS_N_10_SPHVT = '1.1533E-10 * (0.1 / 3 * MC_SPHVT10_CJSWS_N) * PROCESS'
.PARAM DCJSWGS_N_10_SPHVT = '2.8461E-10 * (0.1 / 3 * MC_SPHVT10_CJSWGS_N) * PROCESS'
.PARAM DRDSW_N_10_SPHVT = ' (1.60000E+01 / 3) * MC_SPHVT10_RDSW_N * PROCESS'
.PARAM DU0_N_10_SPHVT = ' (3.80000E-04 / 3) * MC_SPHVT10_U0_N * PROCESS'
.PARAM DWWL_N_10_SPHVT = ' (9.00000E-23 / 3) * MC_SPHVT10_WWL_N * PROCESS'
.PARAM DCGDO_N_10_SPHVT='9.6260E-11*(1/(1+0.0444/3*MC_SPHVT10_TOX_NP)-1)*PROCESS'
.PARAM DCGSO_N_10_SPHVT='9.6260E-11*(1/(1+0.0444/3*MC_SPHVT10_TOX_NP)-1)*PROCESS'
.PARAM DCF_N_10_SPHVT = ' (9.26000E-12 / 3) * MC_SPHVT10_CF_N * PROCESS'

*

.PARAM MC_SPHVT10_CJS_P = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_CJSWS_P = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_CJSWGS_P = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_XW_P = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_XL_P = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_VTH0_P = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_K3_P = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_WLPE0_P = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_LPE0_P = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_WWL_P = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_U0_P = AGAUSS(0, sigma, 3)
.PARAM MC_SPHVT10_CF_P = AGAUSS(0, sigma, 3)

*

.PARAM DTOXE_P_10_SPHVT='2.4500E-09*(0.0408/3*MC_SPHVT10_TOX_NP)*PROCESS'
.PARAM DTOXP_P_10_SPHVT='1.9490E-09*(0.0513/3*MC_SPHVT10_TOX_NP)*PROCESS'
.PARAM DXW_P_10_SPHVT = ' (2.00000E-09 / 3) * MC_SPHVT10_XW_P * PROCESS'
.PARAM DXL_P_10_SPHVT = ' (2.00000E-09 / 3) * MC_SPHVT10_XL_P * PROCESS'
.PARAM DVTH0_P_10_SPHVT = ' (1.50000E-02 / 3) * MC_SPHVT10_VTH0_P * PROCESS'
.PARAM DK3_P_10_SPHVT = ' (2.60000E+01 / 3) * MC_SPHVT10_K3_P * PROCESS'
.PARAM DWLPE0_P_10_SPHVT = ' (4.20000E-09 / 3) * MC_SPHVT10_WLPE0_P * PROCESS'
.PARAM DLPE0_P_10_SPHVT = ' (1.10000E-08 / 3) * MC_SPHVT10_LPE0_P * PROCESS'
.PARAM DCGSL_P_10_SPHVT='1.9430E-10*(1/(1+0.0408/3*MC_SPHVT10_TOX_NP)-1)*PROCESS'
.PARAM DCGDL_P_10_SPHVT='1.9430E-10*(1/(1+0.0408/3*MC_SPHVT10_TOX_NP)-1)*PROCESS'

```

```

.PARAM DCJS_P_10_SPHVT='1.2538E-03*(0.1/3*MC_SPHVT10_CJS_P)*PROCESS'
.PARAM DCJSWS_P_10_SPHVT='1.3013E-10*(0.1/3*MC_SPHVT10_CJSWS_P)*PROCESS'
.PARAM DCJSWGS_P_10_SPHVT='2.6154E-10*(0.1/3*MC_SPHVT10_CJSWGS_P)*PROCESS'
.PARAM DWWL_P_10_SPHVT = ' (4.00000E-21 / 3) * MC_SPHVT10_WWL_P * PROCESS'
.PARAM DU0_P_10_SPHVT = ' (1.40000E-04 / 3) * MC_SPHVT10_U0_P * PROCESS'
.PARAM DCGDO_P_10_SPHVT='4.0150E-11* (1 / (1 + 0.0408/3*MC_SPHVT10_TOX_NP)-1)*PROCESS'
.PARAM DCGSO_P_10_SPHVT='4.0150E-11* (1 / (1 + 0.0408/3*MC_SPHVT10_TOX_NP)-1)*PROCESS'
.PARAM DCF_P_10_SPHVT = ' (9.08000E-12 / 3) * MC_SPHVT10_CF_P * PROCESS'

.SUBCKT MODN D G S B
+ W=0U L=0U MF=1
+ NRD=0 NRS=0 RDC=0 RSC=0
+ DTEMP=0 NF=1 MIS_FLAG=1 SA=0 SB=0 SD=0
+ NF_ODD='NF-2*INT(NF/2)' NF_EVEN='1-NF_ODD'
+ A_UNIT='SD*W/NF' P_UNIT='2*(SD+W/NF)' A_SA_EDGE='SA*W/NF' P_SA_EDGE='2*(SA+W/NF)'
+ A_SB_EDGE='SB*W/NF' P_SB_EDGE='2*(SB+W/NF)'
+ AD='(NF_ODD*(A_UNIT*((NF+1)/2-1)+A_SB_EDGE) + NF_EVEN*(A_UNIT*NF/2))/NF'
+ AS='(NF_ODD*(A_UNIT*((NF+1)/2-1)+A_SA_EDGE)
      + NF_EVEN*(A_UNIT*(NF/2-1)+A_SA_EDGE+A_SB_EDGE))/NF'
+ PD='(NF_ODD*(P_UNIT*((NF+1)/2-1)+P_SB_EDGE) + NF_EVEN*(P_UNIT*NF/2))/NF'
+ PS='(NF_ODD*(P_UNIT*((NF+1)/2-1)+P_SA_EDGE)
      + NF_EVEN*(P_UNIT*(NF/2-1)+P_SA_EDGE+P_SB_EDGE))/NF'

*****
**** MISMATCH ****
*****

.PARAM AVTN = '4.97e-3'
.PARAM CNTN = '1.33e-2'
.PARAM D_VTH0_MA_N_XX = 'AVTN/SQRT(2*W*L*1E12)'
.PARAM D_U0_MA_N_XX = 'CNTN/SQRT(2*W*L*1E12)'
.PARAM P_VTH0_MA_N = 'D_VTH0_MA_N_XX * MC_SPHVT10_VTH0_MA_N * MISMATCH * MIS_FLAG'
.PARAM P_U0_MA_N = 'D_U0_MA_N_XX * MC_SPHVT10_U0_MA_N * MISMATCH * MIS_FLAG'

M1 D G S B N W=W L=L AS=AS AD=AD PS=PS PD=PD NRD=NRD NRS=NRS
+ NF=NF DTEMP=DTEMP RDC=RDC RSC=RSC SA=SA SB=SB SD=SD
*NMOS
.MODEL N NMOS
*****Model Selectors/Controllers*****
+LEVEL          = 5.4000E+01          VERSION      = 4.3000E+00
+BINUNIT        = 1.0000E+00          PARAMCHK     = 1.0000E+00
+MOBMOD         = 0.0000E+00          CAPMOD       = 2.0000E+00

```

```

+IGCMOD      = 1.0000E+00          IGBMOD      = 1.0000E+00
+GEOMOD      = 0.0000E+00          DIOMOD      = 2.0000E+00
+RDSMOD      = 0.0000E+00          RBODYMOD     = 0.0000E+00
+RGATEMOD    = 0.0000E+00          PERMOD      = 1.0000E+00
+ACNQSMOD    = 0.0000E+00          TRNQSMOD    = 0.0000E+00
+RGEOMOD     = 1.0000E+00

****Process Parameters*****
+TOXE        = '2.2500E-09+DTOXE,N_10_SPHVT'    TOXP      = '1.8680E-09+DTOXP,N_10_SPHVT'
+TOXM        = 2.2500E-09          EPSROX     = 3.9000E+00
+XJ          = 1.2000E-07          NGATE      = 1.5000E+20
+NDEP        = 1.5000E+17          NSD        = 8.0000E+20
+RSH         = 8.0000E+00          RSHG       = 8.0000E+00

****Basic Model Parameters*****
+WINT        = 2.7000E-08          LINT       = 2.2640E-08
+VTH0        = '3.2000E-01+DVTH0,N_10_SPHVT+P_VTH0_MA_N/sqrt(MF)'    K1= 3.7880E-01
+K2          = -3.9360E-03          K3= '4.6750E-01+DDK3,N_10_SPHVT'
+K3B         = 3.2940E+00          W0        = 5.0000E-08
+DVT0        = 6.9600E+00          DVT1       = 1.4090E+00
+DVT2        = -1.0300E-02          DVT0W      = 2.2640E-02
+DVT1W       = 9.8430E+05          DVT2W      = -2.9290E-01
+DSUB        = 1.0400E+00          MINV       = -2.6000E-01
+VOFFL       = -3.6140E-09          DVTPO      = 6.8790E-07
+DVTP1       = 1.0000E-01          LPE0       = '4.3400E-08+DDLPE0,N_10_SPHVT'
+LPEB        = 1.2000E-11          VBM        = -3.0000E+00
+PHIN        = 1.2210E-01          CDSC       = 1.0000E-03
+CDSCB       = 0.0000E+00          CDSCD      = 2.0000E-03
+CIT         = 1.0000E-03          VOFF       = -1.2320E-01
+NFACTOR     = 7.5000E-01          ETA0       = 1.0000E-04
+ETAB        = -1.8020E-01          VFB        = -1.0000E+00
+U0          = '(1.7980E-02+DDU0,N_10_SPHVT)*(1-P_U0_MA_N/sqrt(MF))'
+UB          = 2.8520E-18          UC         = 1.3900E-10
+VSAT        = 1.5450E+05          A0         = 1.8250E+00
+AGS         = 1.1330E+00          A1         = 0.0000E+00
+A2          = 1.0000E+00          B0         = 4.0310E-06
+B1          = 4.9470E-05          KETA       = -1.1340E-02
+DWG         = 0.0000E+00          DWB        = 4.7730E-09
+PCLM        = 3.2450E+00          PDIBLC1    = 6.9580E-01
+PDIBLC2     = 1.1130E-03          PDIBLCB    = 5.0000E-01
+DROUT       = 1.5800E+00          PVAG       = 1.2260E+01
+DELTA       = 7.8220E-03          PSCBE1     = 5.5941E+08
+PSCBE2      = 2.9000E-04          FPROUT     = 0.0000E+00
+PDITS       = 0.0000E+00          PDITS      = 0.0000E+00

```

```

+PDITSL      = 1.0000E-10          UA          = -1.9120E-09
*****Parameters for Asymmetric and Bias-Dependent Rds Model*****
+RDSW        = '7.9800E+01+DRDSW_N_10_SPHVT'    RDSWMIN    = 6.3000E+01
+PRWG        = 3.2000E-02          PRWB        = 1.4250E-01
+WR          = 1.0000E+00
*****Impact Ionization Current Model Parameters*****
+ALPHA0       = 5.7860E-04          ALPHA1      = 0.0000E+00
+BETA0        = 2.2510E+01
*****Gate-Induced Drain Leakage Model Parameters*****
+AGIDL        = 2.8460E-08          BGIDL       = 1.5390E+09
+CGIDL        = 3.7557E+00          EGIDL       = 7.3920E-01
*****Gate Dielectric Tunneling Current Model Parameters*****
+TOXREF       = 2.2500E-09          DLCIG       = 2.0350E-08
+AIGBACC      = 1.1689E-02          BIGBACC     = 3.7530E-03
+CIGBACC      = 2.2315E-01          NIGBACC     = 5.0000E+00
+AIGBINV      = 2.0550E-02          BIGBINV     = 6.2780E-03
+CIGBINV      = 1.6980E-03          EIGBINV     = 6.6490E-01
+NIGBINV      = 1.5090E+04          AIGC        = 1.0860E-02
+BIGC         = 1.8970E-03          CIGC        = 3.3220E-02
+AIGSD        = 1.0100E-02          BIGSD       = 3.2410E-04
+CIGSD        = 5.6600E-03          NIGC        = 5.0000E-01
+POXEDGE      = 1.0200E+00          PIGCD       = 1.0000E+00
+NTOX         = 1.0000E+00          LBIGSD      = 7.4750E-06
*****Charge and Capacitance Model Parameters*****
+DLC          = 1.6870E-08          DWC        = -3.0000E-08
+XPART        = 1.0000E+00          CGSO = '9.6260E-11+DCGSO_N_10_SPHVT'
+CGDO        = '9.6260E-11+DCGDO_N_10_SPHVT'    CGBO       = 0.0000E+00
+CGDL        = '1.8420E-10+DCGDL_N_10_SPHVT'    CGSL = '1.8420E-10+DCGSL_N_10_SPHVT'
+CLC         = 7.6330E-08          CLE        = 7.0000E-01
+CF          = '9.2600E-11+DCF_N_10_SPHVT'    CKAPPAS    = 1.0000E+00
+CKAPPAD     = 1.0000E+00          VFBCV      = 0.0000E+00
+ACDE        = 2.2170E-01          MOIN       = 7.8990E+00
+NOFF        = 2.3840E+00          VOFFCV     = -7.9880E-02
*****High-Speed/RF Model Parameters*****
*****Flicker and Thermal Noise Model Parameters*****
+FNOIMOD      = 1.0000E+00          TNOIMOD    = 0.0000E+00
+EF          = 1.0494E+00          NOIA       = 8.4300E+41
+NOIB        = 1.8600E+23          NOIC       = 2.8700E+09
+EM          = 6.3600E+06          NTNOI      = 1.0000E+00
*****Layout-Dependent Parasitics Model Parameters*****
+XL          = '-1.0000E-08+DXL_N_10_SPHVT'    XW = '-0.0000E-00+DXW_N_10_SPHVT'
+DMCG        = 1.6000E-07          DMCI= 1.0000E-07

```

```

*****Asymmetric Source/Drain Junction Diode Model Parameters*****
+JSS      = 7.9580E-07      JSWS      = 5.2430E-13
+JSWGS     = 5.2430E-13     IJTHSFWD = 3.5620E-03
+IJTHSREV  = 1.5390E-03     BVS      = 1.1210E+01
+XJBVS     = 1.0000E+00     PBS      = 6.5000E-01
+CJS       = '1.1830E-03+DCJS_N_10_SPHVT' MJS      = 3.4000E-01
+PBSWS     = 9.9000E-01     CJSWS='1.1533E-10+DCJSWS_N_10_SPHVT'
+MJSWS     = 1.0000E-01     PBSWGS= 9.9000E-01
+CJSWGS    = '2.8461E-10+DCJSWGS_N_10_SPHVT' MJSWGS= 7.8000E-01

*****Temperature Dependence Parameters*****
+TNOM      = 2.5000E+01     KT1      = -2.8580E-01
+KT1L      = -1.8880E-09    KT2      = -3.4200E-02
+UTE       = -1.3340E+00    UA1      = 2.5050E-09
+UB1       = -2.2650E-18    UC1      = 1.6200E-10
+PRT       = -4.5000E+01    AT       = 1.1660E+05
+NJS       = 1.1120E+00    TPB      = 1.2000E-03
+TCJ       = 9.0000E-04     TPBSW    = 1.0000E-04
+TCJSW     = 4.0000E-04     TPBSWG   = 1.8000E-03
+TCJSWG    = 1.4000E-03     XTIS     = 3.0000E+00
+XTID      = 3.0000E+00

*****dW and dL Parameters*****
+LL        = -8.3880E-14    WL        = -6.1460E-15
+LLN       = 7.2310E-01    WLN       = 1.0000E+00
+LW        = 0.0000E+00    WW        = -1.4230E-14
+LWN       = 1.0000E+00    WWN       = 8.9480E-01
+LWL       = -9.8820E-21    WWL       = '1.9700E-21+DWWL_N_10_SPHVT'
+LLC       = -1.1530E-14    WLC       = 0.0000E+00
+LWC       = 0.0000E+00    WWC       = 4.5000E-15
+LWLC      = 0.0000E+00    WWLC      = 0.0000E+00

*****Range Parameters for Model Application*****
+LMIN      = 8.0000E-08     LMAX      = 5.0000E-05
+WMIN      = 1.2000E-07     WMAX      = 1.0000E-04

*****Other Parameters*****
+PVTH0     = -1.0000E-05    PK2       = 0
+LK3       = 4.1270E-01     WK3       = -5.3200E-01
+PK3       = -1.0980E-01     LMINV     = -5.2000E-03
+WDVTP0    = -6.0580E-08    WDVTP1    = 1.0390E-01
+LNFACTOR  = 1.2000E-01     PETAB     = 0
+PU0       = '-1.0000E-05*(1-P_U0_MA_N/sqrt(MF))' LUA      = 9.9000E-13
+PUA       = 8.5800E-12     PUB       = -1.2770E-20
+PVSAT     = -8.5320E+02    LA0       = -2.0000E-01
+PKETA     = -1.0660E-03    PPDIBLC1 = 5.0000E-03

```

```

+WDROUT      = 9.0000E-02          PPVAG      = -4.2000E-02
+LDELTA      = 8.9000E-04          WRDSW      = 1.0000E-10
+PRDSW       = -1.0000E-03         WKT1      = 2.9600E-03
+LKT2        = -1.0000E-03         WUTE      = 5.2070E-02
+LUA1        = -2.6000E-11         PUA1      = 8.9060E-12
+LUB1        = -2.8290E-20         LUC1      = -8.0000E-12
+PAT         = -1.1400E+03         WLPE0     ='0.00E+00+DWLPE0_N_10_SPHVT'
+WAIGC       = 1.6000E-05          WUB       = 8.9000E-21
+WLPEB       = -1.546e-009         PNFACTOR= -2.0000E-02
+PETA0       = 8.8000E-04          LVOFFCV   = -3.9000E-03
+WAO         = 2.0000E-01          WAGS      = 3.0200E-02

*****User Drop Parameters*****
+SAREF       = 1.76E-006           SBREF      = 1.76E-006
+WLOD        = 0                   KVTH0      = 5E-008
+LKVTH0      = 3.9E-006           WKVTH0     = 6E-008
+PKVTH0      = -4e-14             LLODVTH    = 1
+WLODVTH     = 1                   STK2       = 0
+LODK2       = 1                   LODETA0    = 1
+KU0         = -1.65E-008          LKU0       = -1.01E-009
+WKU0        = -4E-008            PKU0       = 1E-015
+LLODKU0     = 1.05               WLODKU0    = 1
+KVSAT       = 0.99               STETA0     = -2.8E-011
+TKU0        = 0

```

.ENDS

```

.SUBCKT MODP D G S B
+ W=0U L=0U MF=1
+ NRD=0 NRS=0 RDC=0 RSC=0
+ DTEMP=0 NF=1 MIS_FLAG=1 SA=0 SB=0 SD=0
+ NF_ODD='NF-2*INT(NF/2)' NF_EVEN='1-NF_ODD'
+ A_UNIT='SD*W/NF' P_UNIT='2*(SD+W/NF)' A_SA_EDGE='SA*W/NF' P_SA_EDGE='2*(SA+W/NF)'
+ A_SB_EDGE='SB*W/NF' P_SB_EDGE='2*(SB+W/NF)'
+ AD='(NF_ODD*(A_UNIT*((NF+1)/2-1)+A_SB_EDGE) + NF_EVEN*(A_UNIT*NF/2))/NF'
+ AS='(NF_ODD*(A_UNIT*((NF+1)/2-1)+A_SA_EDGE)
+   + NF_EVEN*(A_UNIT*(NF/2-1)+A_SA_EDGE+A_SB_EDGE))/NF'
+ PD='(NF_ODD*(P_UNIT*((NF+1)/2-1)+P_SB_EDGE)
+   + NF_EVEN*(P_UNIT*NF/2))/NF'
+ PS='(NF_ODD*(P_UNIT*((NF+1)/2-1)+P_SA_EDGE)
+   + NF_EVEN*(P_UNIT*(NF/2-1)+P_SA_EDGE+P_SB_EDGE))/NF'

```



```

*****
**** MISMATCH ****
*****

.PARAM AVTP = '3.32e-3'
.PARAM CNTP = '1.18e-2'
.PARAM D_VTH0_MA_P_XX = 'AVTP/SQRT(2*W*L*1E12)'
.PARAM D_U0_MA_P_XX = 'CNTP/SQRT(2*W*L*1E12)'
.PARAM P_VTH0_MA_P = 'D_VTH0_MA_P_XX * MC_SPHVT10_VTH0_MA_P * MISMATCH * MIS_FLAG'
.PARAM P_U0_MA_P = 'D_U0_MA_P_XX * MC_SPHVT10_U0_MA_P * MISMATCH * MIS_FLAG'

M1 D G S B P W=W L=L AS=AS AD=AD PS=PS PD=PD NRD=NRD NRS=NRS
+ NF=NF DTEMP=DTEMP RDC=RDC RSC=RSC SA=SA SB=SB SD=SD
.MODEL P PMOS
*****Model Selectors/Controllers*****
+LEVEL      = 5.4000E+01      VERSION      = 4.3000E+00
+BINUNIT     = 1.0000E+00      PARAMCHK    = 1.0000E+00
+MOBMOD      = 0.0000E+00      CAPMOD     = 2.0000E+00
+IGCMOD      = 1.0000E+00      IGBMOD     = 1.0000E+00
+GEOMOD      = 0.0000E+00      DIOMOD     = 2.0000E+00
+RDSDMOD     = 0.0000E+00      RBODYMOD   = 0.0000E+00
+RGATEMOD    = 0.0000E+00      PERMOD     = 1.0000E+00
+ACNQSMOD    = 0.0000E+00      TRNQSMOD   = 0.0000E+00
+RGEOMOD     = 1.0000E+00

*****Process Parameters*****
+TOXE        = '2.4500E-09+DTOXE_P_10_SPHVT'      TOXP='1.9490E-09+DTOXP_P_10_SPHVT'
+TOXM        = 2.4500E-09      EPSROX= 3.9000E+00
+XJ          = 1.2000E-07      NGATE = 1.0000E+20
+NDEP        = 3.0000E+17      NSD   = 6.0000E+20
+RSH         = 8.0000E+00      RSHG  = 8.0000E+00

*****Basic Model Parameters*****
+WINT        = 5.2200E-08      LINT    = -1.2150E-08
+VTH0        = '-2.7300E-01+DVTH0_P_10_SPHVT+P_VTH0_MA_P/sqrt(MF)'      K1=3.2640E-01
+K2          = -3.7280E-02      K3       = '-8.6360E+00+DK3_P_10_SPHVT'
+K3B         = 2.0610E+01      W0        = 2.6180E-06
+DVT0        = 7.0830E-03      DVT1     = 2.5060E-01
+DVT2        = -1.3800E-01      DVT0W    = 7.6250E-04
+DVT1W       = 4.2100E+05      DVT2W    = 1.0000E-02
+DSUB        = 6.9620E-01      MINV     = 1.5000E-01
+VOFFL       = -2.4850E-09      DVTP0    = 2.4730E-07
+DVTP1       = 7.8000E-01      LPE0     = '0.7100E-08+DLPE0_P_10_SPHVT'

```

```

+LPEB      = 5.8520E-08          VBM      = -3.0000E+00
+PHIN      = 1.1420E-01          CDSC     = 1.0000E-03
+CDSCB     = 0.0000E+00          CDSCD   = 0.0000E+00
+CIT       = 0.0000E+00          VOFF     = -1.0540E-01
+NFACTOR    = 7.5650E-01          ETA0    = 3.9980E-01
+ETAB      = -3.5190E-01          VFB     = -1.0000E+00
+U0        = ' (7.2600E-03+DU0_P_10_SPHVT)*(1-P_U0_MA_P/sqrt(MF)) '
+UB        = 1.0900E-18          UC       = -1.5950E-11
+VSAT      = 1.5620E+05          A0      = 2.3520E+00
+AGS       = 1.0850E+00          A1      = 0.0000E+00
+A2        = 3.5000E-01          B0      = 6.5600E-07
+B1        = 4.9610E-05          KETA    = -6.5400E-02
+DWG       = -2.6510E-08          DWB     = -5.8000E-09
+PCLM      = 3.5490E-01          PDIBLC1 = 1.0000E-07
+PDIBLC2   = 1.2320E-03          PDIBLCB = -4.8600E-01
+DROUT     = 5.1190E-01          PVAG    = 0.0000E+00
+DELTA     = 9.9470E-03          PSCBE1  = 1.5350E+09
+PSCBE2    = 1.1467E-07          FPROUT  = 2.7610E+02
+PDITS     = 5.0000E-07          PDITS   = 0.0000E+00
+PDITSL    = 0.0000E+00          UA      = 4.5580E-11

*****Parameters for Asymmetric and Bias-Dependent Rds Model*****
+RDSW      = 1.1800E+02          RDSWMIN = 1.0150E+02
+PRWG      = 0.0000E+00          PRWB    = -8.0000E-02
+WR        = 1.0000E+00

*****Impact Ionization Current Model Parameters*****
+ALPHA0     = 3.0000E-08          ALPHA1  = 0.0000E+00
+BETA0      = 2.1700E+01

*****Gate-Induced Drain Leakage Model Parameters*****
+AGIDL      = 4.2700E-10          BGIDL   = 1.4510E+09
+CGIDL      = 1.1080E+00          EGIDL   = 6.4550E-01

*****Gate Dielectric Tunneling Current Model Parameters*****
+TOXREF     = 2.4500E-09          DLCIG   = 1.8000E-08
+AIGBACC    = 1.0360E-02          BIGBACC = 8.1090E-03
+CIGBACC    = 9.1080E-01          NIGBACC = 4.3790E+00
+AIGBINV    = 1.6100E-02          BIGBINV = 6.2290E-03
+CIGBINV    = 1.0950E-01          EIGBINV = 9.6280E-02
+NIGBINV    = 3.0500E+00          AIGC    = 5.9290E-03
+BIGC       = 5.1880E-04          CIGC    = 5.8650E-02
+AIGSD      = 5.9900E-03          BIGSD   = 9.9050E-05
+CIGSD      = 3.4690E-06          NIGC    = 1.3800E+00
+POXEDGE    = 1.0000E+00          PIGCD   = 1.0000E+00
+NTOX       = 1.0000E+00

```

```

****Charge and Capacitance Model Parameters*****
+DLC          = 4.0050E-09                DWC      = -3.4160E-08
+XPART        = 1.0000E+00                CGSO=' 4.0150E-11+DCGSO_P_10_SPHVT'
+CGDO        = ' 4.0150E-11+DCGDO_P_10_SPHVT'  CGBO      = 0.0000E+00
+CGDL        = ' 1.9430E-10+DCGDL_P_10_SPHVT'  CGSL=' 1.9430E-10+DCGSL_P_10_SPHVT'
+CLC          = 1.0330E-07                CLE       = 5.7730E-01
+CF           = ' 9.0800E-11+DCF_P_10_SPHVT'  CKAPPAS= 1.0000E+00
+CKAPPAD      = 1.0000E+00                VFBCV     = -1.0000E+00
+ACDE         = 3.6180E-01                MOIN      = 8.4860E+00
+NOFF         = 2.1890E+00                VOFFCV    = -4.4100E-02

****High-Speed/RF Model Parameters*****
****Flicker and Thermal Noise Model Parameters*****
+FNOIMOD      = 1.0000E+00                TNOIMOD   = 0.0000E+00
+EF           = 1.0926E+00                NOIA      = 3.9700E+42
+NOIB         = 3.6300E+23                NOIC      = 1.9900E+10
+EM           = 4.1000E+07                NTNOI     = 1.0000E+00

****Layout-Dependent Parasitics Model Parameters*****
+XL           = '-1.0000E-08+DXL_P_10_SPHVT'  XW =' -0.00E-00+DXW_P_10_SPHVT'
+DMCG         = 1.6000E-07                DMCI      = 1.0000E-07

****Asymmetric Source/Drain Junction Diode Model Parameters*****
+JSS          = 1.6920E-07                JSWS      = 1.1100E-13
+JSWGS        = 1.1100E-13                IJTHSFWD  = 3.4580E-03
+IJTHSREV     = 1.9210E-03                BVS       = 8.8630E+00
+XJBVS        = 1.0000E+00                PBS       = 7.0000E-01
+CJS          = ' 1.2538E-03+DCJS_P_10_SPHVT'  MJS       = 3.0000E-01
+PBSWS        = 9.9000E-01                CJSWS=' 1.3013E-10+DCJSWS_P_10_SPHVT'
+MJSWS        = 1.0000E-01                PBSWGS    = 9.6000E-01
+CJSWGS       = ' 2.6154E-10+DCJSWGS_P_10_SPHVT'  MJSWGS= 9.8900E-01

****Temperature Dependence Parameters*****
+TNOM         = 2.5000E+01                KT1       = -2.3040E-01
+KT1L        = -6.9600E-09                KT2       = -6.2020E-02
+UTE         = -1.4060E+00                UA1       = -1.3050E-10
+UB1         = 5.1000E-19                UC1       = 0.0000E+00
+PRT         = -1.0000E+02                AT        = 1.9040E+04
+NJS         = 1.0542E+00                TPB       = 1.6000E-03
+TCJ         = 9.0000E-04                TPBSW     = 1.0000E-04
+TCJSW       = 4.0000E-04                TPBSWG    = 1.4000E-03
+TCJSWG      = 1.8000E-03                XTIS      = 3.0000E+00
+XTID        = 3.0000E+00

****dW and dL Parameters*****
+LL          = 3.4010E-14                WL        = -6.0000E-16
+LLN         = 7.6500E-01                WLN       = 1.0000E+00

```

```

+LW      = -4.0000E-15      WW      = -1.2840E-12
+LWN      = 1.0000E+00      WWN      = 6.7930E-01
+LWL      = 0.0000E+00      WWL      = '-1.1530E-20+DWL_P_10_SPHVT'
+LLC      = 3.3010E-14      WLC      = 0.0000E+00
+LWC      = 0.0000E+00      WWC      = 2.0000E-13
+LWLC     = 0.0000E+00      WWLC     = 0.0000E+00

*****Range Parameters for Model Application*****
+LMIN      = 8.0000E-08      LMAX      = 5.0000E-05
+WMIN      = 1.2000E-07      WMAX      = 1.0000E-04

*****Other Parameters*****
+PVTH0     = -1.7080E-03      WK3       = -6.9340E+00
+WK3B      = 5.3770E+00      PK3B      = -3.1550E-01
+LDVT0     = 3.7540E-02      PDSUB     = -3.6680E-03
+LLPE0     = 4.6200E-09      WLPE0     = '-8.4620E-09+DWLPE0_P_10_SPHVT'
+WLPB      = -1.0720E-08      WVOFF     = 7.6650E-04
+LNFACTOR  = 3.9000E-02      WNFACTOR  = -1.0650E-01
+WUA       = -5.8690E-11      WUB       = 1.2300E-19
+PUB       = -5.3250E-21      PVSAT     = -1.7380E+03
+LA0       = 1.2610E-01      LAGS      = 5.0000E-01
+LKETA     = 4.4490E-03      WKETA     = 8.8800E-03
+PPCLM     = 6.7610E-03      LDELTA    = 9.0000E-04
+PRDSW     = 1.0200E-01      WKT1      = 2.0000E-03
+LKT2      = 1.9700E-03      WUTE      = -2.8000E-02
+WUA1      = -3.8400E-11      LUB1      = -5.0600E-20
+WUC1      = 4.3440E-11      PETAB     = 0.01
+LVOFFCV   = -4.0000E-03      WAIGSD    = 3.3000E-05
+PAIGSD    = 4.5000E-06      PNFACTOR  = 1.8000E-02
+SAREF     = 1.76E-006      SBREF     = 1.76E-006
+WLOD      = 0              KVTH0     = -8E-10
+LKVTH0    = -1.5e-6       WKVTH0    = 6e-7
+PKVTH0    = 0              LLODVTH   = 0.8
+WLODVTH   = 1              STK2      = 0
+LODK2     = 1              LODETA0   = 1
+KU0       = 5.3E-007      LKU0      = 5.8E-004
+WKU0      = -1.10E-9      PKU0      = -2.5E-010
+LLODKU0   = 0.68         WLODKU0   = 0.85
+KVSAT     = 1              STETA0    = 3.8E-010
+TKU0      = 0
.ENDS

```


Bibliography

- [1] F. Medeiro, F.V. Fernández, R. Domínguez-Castro, and A. Rodríguez-Vázquez. A Statistical Optimization-based Approach For Automated Sizing Of Analog Cells. *International Conference on Computer-Aided Design*, pages 594 – 597, 1994.
- [2] John R. Koza, Lee W. Jones, Martin A. Keane, Matthew J. Streeter, and Sameer H. Al-Sakran. *Genetic Programming Theory and Practice II*, chapter Toward Automated Design Of Industrial-Strength Analog Circuits By Means Of Genetic Programming. Springer, 2005.
- [3] Georges Gielen and Wim Dehaene. Analog and digital circuit design in 65 nm CMOS: end of the road? *Design, Automation and Test in Europe*, pages 37–42, 2005.
- [4] Georges Gielen, Tom Eeckelaert, Ewout Martens, and Trent McConaghy. Automated synthesis of complex analog circuits. *European Conference on Circuit Theory and Design*, 27(30):20–23, August 2007.
- [5] Guo Yu and Peng Li. Yield-Aware Analog Integrated Circuit Optimization Using Geostatistics Motivated Performance Modeling. *IEEE/ACM International Conference on Computer-Aided Design*, pages 464–469, November 2007. San Jose, C.A.
- [6] G. Yu and Peng Li. Yield-Aware Hierarchical Optimization of Large Analog Integrated Circuits. *IEEE/ACM International Conference on Computer-Aided Design*, pages 79 – 84, November 2008.
- [7] Tom Eeckelaert, Raf Schoofs, Georges Gielen, Michiel Steyaert, and Willy Sansen. Hierarchical Bottom-up Analog Optimization Methodology Validated by a Delta-Sigma A/D Converter Design for the 802.11 a/b/g Standard. *Design Automation Conference*, pages 25 – 30, July 2006.
- [8] T. Eeckelaert, T. McConaghy, and G. Gielen. Efficient multiobjective synthesis of analog circuits using hierarchical pareto optimal performance hypersurfaces. *Design, Automation and Test in Europe*, 2:1070 – 1075, March 2005.
- [9] M. Bühler, J. Koehl, J. Bickford, J. Hibbeler, U. Schlichtmann, R. Sommer, M. Pronath, and A. Ripp. DFM/DFY Design for Manufacturability and Yield - influence of process variations in digital, analog and mixed-signal circuit design. *Design, Automation, and Test in Europe*, pages 387–392, 2006. Munich, Germany.
- [10] Esteban Tlelo-Cuautle, Ivick Guerra-Gómez, Carlos Alberto Reyes-García, and Miguel Aurelio Duarte-Villaseñor. *Intelligent Systems for Automated Learning and Adaptation: Emerging Trends and Applications*, chapter 8: Synthesis of Analog Circuits by Genetic Algorithms and their Optimization by Particle Swarm Optimization, pages 173–192. IGI Global, 2009.

- [11] Yann Cooren, Mourad Fakhfakh, Mourad Loulou, and Patrick Siarry. Optimizing Second Generation Current Conveyors using Particle Swarm Optimization. *Internatonal Conference on Microelectronics*, pages 365–368, December 2007. Cairo.
- [12] Jernej Olenšek, Árpád Bűrmen, Janez Puhan, and Tadej Tuma. DESA: a new hybrid global optimization method and its application to analog integrated circuit sizing. *Journal of Global Optimization*, 44(1):53–77, May 2009.
- [13] Bo Liu, Yan Wang, Zhiping Yu, Leibo Liu, Miao Li, Zheng Wang, Jing Lu, and Francisco V. Fernndez. Analog Circuit Optimization System Based on Hybrid Evolutionary Algorithms. *Integration, the VLSI Journal*, 42(2):137–148, February 2009.
- [14] Ivick Guerra-Gómez, Esteban Tlelo-Cuautle, Trent McConaghy, and Geroges Gielen. Decomposition-Based Multi-Objective Optimization of Second-Generation Current Conveyors. *IEEE International Midwest Symposium on Circuits and Systems*, pages 220–223, August 2009.
- [15] Anthony Scanlan. Application of Genetic Algorithms to Analogue Circuit Synthesis. *Irish Signals and Systems Conference*, pages 339–344, June 2006. Dublin.
- [16] Raf Schoofs, Tom Eeckelaert, Michiel Steyaert, Goerges Gielen, and Willy Sansen. A Continuous-Time Delta-Sigma Modulator for 802.11a/b/g WLAN Implemented with a Hierarchical Bottom-up Optimization Methodology. *IEEE International Conference on Electronics, Circuits and Systems*, 10(13):950 – 953, December 2006.
- [17] Tom Eeckelaert, Raf Schoofs, Georges Gielen, Michiel Steyaert, and Willy Sansen. An Efficient Methodology for Hierarchical Synthesis of Mixed-Signal Systems with Fully Integrated Building Block Topology Selection. *Design, Automation and Test in Europe Conference and Exhibition*, pages 1 – 6, April 2007.
- [18] Henry Chang, Alberto Snagiovanni-Vincentelli, Felice Balarin, Edoardo Charbon, Umakanta Choudhury, Gani Jusuf, Liu Edward, Enrico Malavasi, Robert Neff, and Paul R. Gray. A Top-Down, Constraint-Driven Design Methodology for Analog Integrated Circuits. *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 8.4.1 – 8.4.6, 1992.
- [19] Geroges Gielen, Trent McConaghy, and Tom Eeckelaert. Performance Space Modeling for Hierarchical Synthesis of Analog Integrated Circuits. *Design Automation Conference*, pages 881–886, June 2005. Anaheim, California, USA.
- [20] Saurabh K Tiwary, Pragati K Tiwary, and Rob A Rutenbar. Generation of Yield-Aware Pareto Surfaces for Hierarchical Circuit Design Space Exploration. *Design Automation Conference*, pages 31–36, July 2006. San Francisco, California, USA.
- [21] Rob A. Rutenbar. Design Automation for Analog: The Next Generation of Tool Challenges. *IEEE/ACM International Conference on Computer-Aided Design*, 5(9):458 – 460, November 2006.
- [22] Trent McConaghy, Pieter Palmers, Georges Gielen, and Michiel Steyaert. Automated extraction of expert knowledge in analog topology selection and sizing. *International Conference on Computer-Aided Design*, pages 392–395, November 2008. San Jose, CA.
- [23] Ewout Martens and Georges Gielen. Classification of analog synthesis tools based on their architecture selection mechanisms. *INTEGRATION, the VLSI journal*, 41:238–252, February 2008.

- [24] T. McConaghy and G. Gielen. CAFFEINE: template-free symbolic model generation of analog circuits via canonical form functions and genetic programming. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(8):1162 – 1175, 2009.
- [25] Kurt J. Antreich, Peter Leibner, and Fritz Pörnbacher. Nominal design of integrated circuits on circuit level by an interactive improvement method. *IEEE Transactions on Circuits and Systems*, 35(12):1501–1511, December 1988.
- [26] Xu Yang, Li Xin, Hsiung Kan-Lin, S. Boyd, and I. Nausieda. OPERA: optimization with ellipsoidal uncertainty for robust analog IC design. *Design Automation Conference*, pages 632 – 637, 2005.
- [27] R. Phelps, M. Krasnicki, R.A. Rutenbar, L.R. Carley, and J.R. Hellums. Anaconda: simulation-based synthesis of analog circuits via stochastic pattern search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(6):703–717, June 2000.
- [28] F. Madeiro, R. Rodriguez-Macías, F.V. Fernández, R. Domínguez-Castro, J.L. Huertas, and A. Rodríguez-Vázquez. Global Design of Analog Cells using Statistical Optimization Techniques. *Analog Integrated Circuits and Signal Processing*, 6(3):179–195, 1994.
- [29] R. Castro-López, E. Roca, and F.V. Fernández. Multimode Pareto fronts for design of reconfigurable analogue circuits. *Electronics Letters*, 45(2):95–96, January 2009.
- [30] Trent McConaghy and Georges Gielen. Analysis of Simulation-Driven Numerical Performance Modeling Techniques for Application to Analog Circuit Optimization. *IEEE International Symposium on Circuits and Systems*, 2:1298–1301, May 2005.
- [31] M.G.R. Degrauwe, O. Nys, E. Dijkstra, J. Rijmenants, S. Bitz, B.L.A.G. Goffart, E.A. Vittoz, S. Cserveny, C. Meixenberger, G. van der Stappen, and H.J. Oguey. IDAC: an interactive design tool for analog CMOS circuits. *IEEE Journal of Solid-State Circuits*, 22(6):1106 – 1116, 1987.
- [32] B.J. Sheu, A.H. Fung, and Y.-N. Lai. A knowledge-based approach to analog IC design. *IEEE Transactions on Circuits and Systems*, 35(2):256 – 258, February 1988.
- [33] R. Harjani. OASYS: a framework for analog circuit synthesis. *Proceedings ASIC Seminar and Exhibit*, pages P13 – 1/1–4, 1989.
- [34] P. Heikkilä, M. Valtonen, and H. Pohjonen. Automated dimensioning of MOS transistors without using topology-specific explicit formulas. *IEEE International Symposium on Circuits and Systems*, 2:1131 – 1134, 1989.
- [35] Fatehy El-Turky and Elizabeth E. Perry. BLADES: an artificial intelligence approach to analog circuit design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(6):680–692, June 1989.
- [36] Georges Gielen, H.C.C. Walscharts, and Willy Sansen. ISAAC: a symbolic simulator for analog integrated circuits. *IEEE Journal of Solid-State Circuits*, 24(6):1587–1597, December 1989.
- [37] Antony H. Fung, Bang W. Lee, and Bing J. Sheu. Self-reconstructing technique for expert system-based analog IC designs. *IEEE Transactions on Circuits and Systems*, 36(2):318–321, February 1989.
- [38] Georges Gielen, Herman C. C. Walscharts, and Willy M. Sansen. Analog circuit design optimization based on symbolic simulation and simulated annealing. *IEEE Journal of Solid-State Circuits*, 25(3):707–713, June 1990.

- [39] Zhen-Qiu Ning, Ton Mouthaan, and Hans Wallinga. SEAS: a simulated evolution approach for analog circuit synthesis. *Proceedings of the IEEE, Custom Integrated Circuits Conference*, pages 5.2– 1–4, May 1991.
- [40] Kurt J. Antreich, Helmut E. Graeb, and Claudia U. Wieser. Circuit Analysis and Optimization Driven by Worst-case Distances. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(1):57–71, January 1994.
- [41] W. Tian, Michael and C.-J. Richard Shi. Worst Case Tolerance Analysis of Linear Analog Circuits Using Sensitivity Bands. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 47(8):1138–1145, August 2000.
- [42] E.S. Ochotta, R.A. Rutenbar, and L.R. Carley. Synthesis of High-Performance Analog Circuits in ASTR-WOBLX. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(3):273 – 294, March 1996.
- [43] Suresh Seshadri and Jacob A. Abraham. Frequency Response Verification of Analog Circuits Using Global Optimization Techniques. *Journal of Electronic Testing: Theory and Applications*, 17(5):395–408, October 2001.
- [44] Bart De Sm and Georges Gielen. HOLMES: Capturing the YieldOptimized Design Space Boundaries of Analog and RF Integrated Circuits. *Design, Automation and Test in Europe Conference and Exhibition*, pages 256–261, 2003.
- [45] B. De Smedt and G.G.E. Gielen. WATSON: design space boundary exploration and model generation for analog and RFIC design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(2):213 – 224, 2003.
- [46] Wim Kruiskamp and Domine Leenaerts. DARWIN: CMOS opamp Synthesis by Means of a Genetic Algorithm. *Conference on Design Automation*, pages 433 – 438, 1995.
- [47] P.C. Maulik, L.R. Carley, and R.A. Rutenbar. Integer programming based topology selection of cell-level analog circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(4):401 – 412, 1995.
- [48] Trent McConaghy, Pieter Palmers, Georges Gielen, and Michiel Steyaert. Simultaneous Multi-Topology Multi-Objective Sizing Across Thousands of Analog Circuit Topologies. *Design Automation Conference*, 4(8):944 – 947, June 2007. San Diego, CA, USA.
- [49] Carlo Guardiani, Sharad Saxena, Patrick McNamara, Phillip Schumaker, and Dale Coder. An Asymptotically Constant, Linearly Bounded Methodology for the Statistical Simulation of Analog Circuits Including Component Mismatch Effects. *Design Automation Conference*, pages 15–18, June 2000. Los Angeles, California.
- [50] K. Antreich, J. Eckmueller, H. Graeb, M. Pronath, F. Schenkel, R. Schwencker, and S. Zizala. WiCkeD: Analog Circuit Synthesis Incorporating Mismatch. *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 511 – 514, May 2000.
- [51] Rob A. Rutenbar, Georges Gielen, and Jaijeet Roychowdhury. Hierarchical Modeling Optimization, and Synthesis for System-Level Analog and RF Designs. *Proceedings of the IEEE*, 95(3):640–669, March 2007.
- [52] Liu Bo, F.V. Fernandez, and G.G.E. Gielen. Efficient and Accurate Statistical Analog Yield Optimization and Variation-Aware Circuit Sizing Based on Computational Intelligence Techniques. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(6):793 – 805, 2011.

- [53] Virtuoso NeoCircuit. 2007.
- [54] Synopsys Circuit Explorer. 2008.
- [55] munEDA. 2009.
- [56] Magma Titan, 2009.
- [57] Arsyn Automated Design and Reuse Platform. 2009.
- [58] Trent McConaghy and Georges Gielen. Automation in Mixed-Signal Design: Challenges and Solutions in the Wake of the Nano Era. *IEEE/ACM International Conference on Computer-Aided Design*, 5(9):461 – 463, November 2006.
- [59] Carlo Roma, Pierluigi Daglio, Guido De Sandre, Marco Pasotti, and Marco Poles. How circuit analysis and yield optimization can be used to detect circuit limitations before silicon results. *International Symposium on Quality of Electronic Design*, pages 107–112, March 2005.
- [60] Carlos A. Coello-Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2nd edition, 2007.
- [61] T. Bäck, D. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation*. IOP Publishing Ltd. and Oxford University Press., 1997.
- [62] Kaisa Miettinen. *Evolutionary Multi-Criterion Optimization*. Springer Berlin / Heidelberg, 2001.
- [63] Kaisa M. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.
- [64] Oliver Schütze, Marco Laumanns, Carlos A. Coello Coello, Michael Dellnitz, and El-Ghazali Talbi. Convergence of stochastic search algorithms to finite size pareto set approximations. *Journal of Global Optimization*, 41(4):559–577, 2008.
- [65] K Deb. *Multi-objective optimization using evolutionary algorithms*. New York: Wiley, 2001.
- [66] Qingfu Zhang and Hui Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, December 2007.
- [67] Gilles Brassard and Paul Bratley. *Fundamentals of Algorithmics*. Prentice Hall, 1996.
- [68] Bruno R. Preiss. *Data Structures and Algorithms with Object-Oriented Design Patterns in C++*. John Wiley & Sons, August 1998.
- [69] J . H. Holland. *Adaptation in Natural and Artificial Systems*. Technical report, 1985.
- [70] K. Deb and R. Agrawal. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 2005.
- [71] Rainer Storn and Kenneth Price. Minimizing the real functions of the ICEC’96 contest by Differential Evolution. *IEEE International Conference on Evolutionary Computation*, pages 842–844, 1996.
- [72] I. Guerra-Gomez, E. Tlelo-Cuautle, T. McConaghy, and G. Gielen. Optimizing current conveyors by evolutionary algorithms including differential evolution. *IEEE International Conference on Electronics, Circuits, and Systems*, pages 259 – 262, 2009.
- [73] Rainer Storn. *Diferential Evolution Research – Trends and Open Questions in Advances in Differential Evolution*. 2008.

- [74] Sunith Bandaru, Rupesh Tulshyan, and Kalyanmoy Deb. Modified SBX and Adaptive Mutation for Real World Single Objective Optimization. *IEEE Congress on Evolutionary Computation*, pages 1335 – 1342, June 2011.
- [75] Kalyanmoy Deb and Mayank Goyal. A Combined Genetic Adaptive Search (GeneAS) for Engineering Design. *Computer Science and Informatics*, 26(4):30–45, 1996.
- [76] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.
- [77] Li Hui and Zhang Qingfu. Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation*, 13(2):284 – 302, April 2009.
- [78] J. Kennedy and R. Eberhart. Particle swarm optimization. *IEEE International Conference on Neural Networks*, 4:1942 – 1948, 1995.
- [79] C.A. Coello Coello and M.S. Lechuga. MOPSO: a proposal for multiple objective particle swarm optimization. *Congress on Evolutionary Computation*, 2:1051 – 1056, 2002.
- [80] E Zitzler, K Deb, and L Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *IEEE Trans. Evolutionary Computation*, 8(2):173–195, 2002.
- [81] *HSPICE_{TM} Simulation and Analysis User Guide*. Synopsys, 2008.
- [82] E. Tlelo-Cuautle, I. Guerra-Gómez, C. A. Reyes-García, and M. A. Duarte-Villaseñor. *Synthesis of Analog Circuits by Genetic Algorithms and their Optimization by Particle Swarm Optimization*, chapter 8, pages 173–192. IGI Global, 2010.
- [83] E. Tlelo-Cuautle, D. Moro-Frías, C. Sánchez-López, and M. A. Duarte-Villaseñor. Synthesis of CCII-s by superimposing VFs and CFs through genetic operations. *IEICE Electronics Express*, 5(11):411– 417, June 2008.
- [84] V. Boos, J. Nowak, M. Sylvester, S. Henker, S. Höppner, H. Grimm, D. Krausse, and R. Sommer. Strategies for initial sizing and operating point analysis of analog circuits. *Design, Automation and Test in Europe Conference and Exhibition*, pages 1 – 3, 2011.
- [85] D. Torres-Munoz and E. Tlelo-Cuautle. Automatic biasing and sizing of CMOS analog integrated circuits. *Midwest Symposium on Circuits and Systems*, 1:915 – 918, Aug 2005.
- [86] F. Leyn, G. Gielen, and W. Sansen. An efficient DC root solving algorithm with guaranteed convergence for analog integrated CMOS circuits. *International Conference on Computer-Aided Design*, pages 304–307, 1998.
- [87] F. Javid, R. Iskander, and M.-M. Louerat. Simulation-based hierarchical sizing and biasing of analog firm IPs. *Behavioral Modeling and Simulation Workshop*, pages 43–48, September 2009.
- [88] F. De Bernardinis, P. Nuzzo, and A. Sangiovanni Vincentelli. Mixed signal design space exploration through analog platforms. *Design Automation Conference*, pages 875 – 880, June 2005.
- [89] Mengmeng Ding and Ranga Vemuri. A combined feasibility and performance macromodel for analog circuits. *Design Automation Conference*, pages 63–68, 2005.
- [90] W. Daems, G. Gielen, and W. Sansen. Simulation-based generation of posynomial performance models for the sizing of analog integrated circuits. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 22:517–534, 2003. 5.

- [91] T. McConaghy, P. Palmers, M. Steyaert, and G.G.E. Gielen. Variation-Aware Structural Synthesis of Analog Circuits via Hierarchical Building Blocks and Structural Homotopy. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(9):1281 – 1294, 2009.
- [92] Varun Aggarwal and Una-May O'Reilly. Design of Posynomial Models for Mosfets: Symbolic Regression Using Genetic Algorithms. In Rick L. Riolo, Terence Soule, and Bill Worzel, editors, *Genetic Programming Theory and Practice IV*, volume 5 of *Genetic and Evolutionary Computation*, chapter 7. Springer, 2006.
- [93] D.M. Binkley, C.E. Hopper, S.D. Tucker, B.C. Moss, J.M. Rochelle, and D.P. Foty. A CAD methodology for optimizing transistor current and sizing in analog CMOS design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(2):225 – 237, 2003.
- [94] Danica Stefanovic, Maher Kayal, and Marc Pastre. PAD: A New Interactive Knowledge-Based Analog Design Approach. *Analog Integrated Circuits and Signal Processing*, 42(3):291–299, March 2005.
- [95] E. Tlelo-Cuautle and L. A. Sarmiento-Reyes. Biasing analog circuits using the nullor concept. *Southwest Symposium on Mixed-Signal Design (SSMSD) 2000*, pages 27–30, 2000.
- [96] Leon O. Chua and Pen-Min Lin. Computer-aided analysis of electronic circuits. *Prentice-Hall*, page 140, 1975.
- [97] Zhanhai Qin, Sheldon X. D. Tan, and Chung-Kuan Cheng. Symbolic Analysis and Reduction of VLSI Circuits. *Springer*, pages 155 – 159, 2005.
- [98] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. Introduction to Algorithms. *Mc. Graw Hill*, pages 469 – 472, 2000.
- [99] Rida S. Assaad and Jose Silva-Martinez. The Recycling Folded Cascode: A General Enhancement of the Folded Cascode Amplifier. *IEEE Journal of Solid-State Circuits*, 44(9):2535–2542, September 2009.
- [100] Robert Spence and Randeep Singh Soin. *Tolerance design of electronic circuits*. World Scientific, 1997.
- [101] T. McConaghy, P. Palmers, M. Steyaert, and G.G.E. Gielen. *Variation-Aware Analog Structural Synthesis*. Springer, 2009.
- [102] Sun Wei, R.M.M. Chen, and Jiang Yao-Lin. Tolerance Analysis for Electronic Circuit Design Using the Method of Moments. *IEEE International Symposium on Circuits and Systems.*, 1:565 – 568, 2002.
- [103] Zhigang Hao, S.X.-D. Tan, Shen Ruijing, and Shi Guoyong. Performance Bound Analysis of Analog Circuits Considering Process Variations. *IEEE Design Automation Conference*, pages 310 – 315, June 2011. San Diego, CA.
- [104] I. Guerra-Gómez, E. Tlelo-Cuautle, and L.G. de la Fraga. Sensitivity analysis in the optimal sizing of analog circuits by evolutionary algorithms. *International Conference on Electrical Engineering Computing Science and Automatic Control*, pages 381–385, 2010.
- [105] T.B. Tarim, M. Ismail, and H.H. Kuntman. Robust Design and Yield Enhancement of Low-Voltage CMOS Analog Integrated Circuits. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 48(4), 2001.
- [106] Trent McConaghy and Georges Gielen. Globally reliable variation-aware sizing of analog integrated circuits via response surfaces and structural homotopy. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(11):1627–1640, November 2009.

- [107] H. Graeb, D. Mueller, and U. Schlichtmann. Pareto optimization of analog circuits considering variability. *International Journal of Circuit Theory and Applications*, 37:283–299, 2009.
- [108] D. Mueller-Gritschneider, H. Graeb, and U. Schlichtmann. A successive approach to compute the bounded Pareto front of practical multi-objective optimization problems. *SIAM J. Optimization*, 20:915–934, 2009.
- [109] E. Tlelo-Cuautle, I. Guerra-Gómez, Miguel Aurelio Duarte-Villaseñor, L. G. de la Fraga, G. Flores-Becerra, G. Reyes-Salgado, Carlos Alberto Reyes-García, and G. Rodríguez-Gómez. Applications of evolutionary algorithms in the design automation of analog integrated circuits. *Journal of Applied Sciences*, 10(17):1859–1872, 2010.
- [110] J. Munyakazi and K. Patidar. On Richardson extrapolation for fitted operator finite difference methods. *Applied Mathematics and Computation*, 201(1):465–480, July 2008.
- [111] I. Farago, A. Havasi, and Z. Zlatev. Efficient implementation of stable Richardson Extrapolation algorithms. *Computers and Mathematics with Applications*, 60(8):2309–2325, October 2010.
- [112] Chien-Cheng Tseng and Su-Ling Lee. Digital IIR integrator design using Richardson extrapolation and fractional delay. *IEEE Trans on Circuits and Systems*, 55(8):2300–22309, September 2008.
- [113] A. Soroushian, P. Wriggers, and J. Farjoodi. Asymptotic upper bounds for the errors of Richardson extrapolation with practical application in approximate computations. *International Journal for Numerical Methods in Engineering*, 80(5):565–595, October 2009.
- [114] P. Mohan. Sensitivity Analysis of Third and Fourth-Order Filters. *Circuits, Systems, and Signal Processing*, 29(5):999–1005, 2009.
- [115] M. S. Ghausi and K. R. Laker. *Modern Filter Design*. Prentice Hall, 1981.
- [116] Germund Dahlquist, Åke Björk, and Ned Anderson. *Numerical Methods*. Prentice Hall, 1974.
- [117] Chen Hsiao-Chang, Chen Chun-Hung, and E. Yucesan. Computing efforts allocation for ordinal optimization and discrete event simulation. *IEEE Transactions on Automatic Control*, 45(5):960 – 964, 2000.
- [118] Chun-Hung Chen, Jianwu Lin, Enver Ycesan, and Stephen E. Chick. Simulation Budget Allocation for Further Enhancing the Efficiency of Ordinal Optimization. *Discrete Event Dynamic Systems: Theory and Applications*, 10(3):251–270, 2000.