CIRCUIT LEVEL ANALOG DESIGN AUTOMATION

by

ÖZSUN SERKAN SÖNMEZ

BS, Electrical and Electronics Engineering, Boğaziçi University, 2000

MS, Electrical and Electronics Engineering, Boğaziçi University, 2003

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University
2010

ACKNOWLEDGEMENTS

I would like to express my gratitute to my thesis supervisor Prof. Günhan Dündar for his encouragement, continuous support, valuable guidance and endless patience.

I would like to thank Prof. Georges Gielen for taking time to attend my defense and for commenting on the dissertation, which helped to improve my thesis.

I would like to thank Prof. Ömer Cerid for sharing his broad analog design knowledge throughout the thesis progress. His contribution was crucial especially for testing the design automation system. I would like to thank Prof. H. Levent Akın for his guidance and recommendations regarding optimization techniques.

I would like to thank Assoc. Prof. Arda Deniz Yalçınkaya for his comments, corrections and recommendations regarding the thesis contents and its documentation. I would like to thank Prof. Tülay Yıldırım for attending my thesis defense and for her valuable comments.

I would like to thank my dear friend Balkır Kayaaltı for every moment we shared throughout our fourteen years of under-graduate, MS and PhD studies. I would also like to thank Olcay Durul Azeri and Ahmet Unutulmaz for their help in integrating their thesis studies with mine. I am thankful to the BETA members, especially Okan Zafer Batur and Seyrani Korkmaz, for their help.

Finally, I would like to thank all of the members of my family, whose continuous support and patience made the completion of this thesis possible.

ABSTRACT

CIRCUIT LEVEL ANALOG DESIGN AUTOMATION

This thesis presents a simulation-based analog circuit synthesis methodology, its integration with system and layout level analog synthesis tools, and synthesis examples that were performed to validate the usefulness of the methodology. Simulation-based approach is preferred so that the synthesizer, SACSES, is topology independent. Instead of using a commercially available simulator, an accelerated simulator, SPASE, is implemented. SPASE has various acceleration mechanisms for DC, AC and noise simulation, detailed in the thesis. The search algorithm used is an ES algorithm modified so as to use Metropolis criterion as the selection method. By adding hill-climbing capability similar to simulated annealing, this modification provides a proper balance between local and global search and eliminates the premature convergence problem of GA and ES-based algorithms. Designer effort is minimized with the automated determination and self-evolution of search parameters. Smooth penalty mechanisms for biasing constraints are proposed and embedded in the algorithm. Yield-aware synthesis is performed by utilizing piecewise cubic Hermite splines for response surface modeling. A hierarchical synthesis structure is proposed for integrating SACSES with system level synthesis tools. The hierarchical scheme eliminates the need for extra tools to link levels. SACSES and the layout level synthesis tool is integrated with a feedback loop, by which the effects of layout parasitics on circuit performance are minimized.

ÖZET

DEVRE DÜZEYİNDE ANALOG TASARIM OTOMASYONU

Bu tezde, benzetimli devre sentezi için benzetim-tabanlı bir yöntemler dizisi, bu yöntemler dizisinin sistem ve serim düzeylerindeki sentez araçları ile birleştirilmesi ve yöntemler dizisinin faydalılığını onaylamak amacıyla gerçekleştirilmiş olan sentez sonuçları sunulmaktadır. Benzetim-tabanlı yaklaşımın tercih edilmesi sonucunda sentezleyici, SACSES, ilingeden bağımsız olma özelliğini kazanmıştır. Ticari bir benzeticinin kullanılması yerine hızlandırılmış bir benzetici, SPASE, gerçekleştirilmiştir. SPASE'in DA, AA ve gürültü benzetimleri için detayları bu tezde verilmiş olan çeşitli hızlandırma düzenekleri vardır. Kullanılan arama algoritması, Metropolis ölçütünü seçim yöntemi olarak kullanacak şekilde değiştirilmiş bir ES algoritmasıdır. Bu değişiklik, arama algoritmasına benzetimli tavlama yönteminde olduğu gibi tırmanma yeteneği kazandırarak yerel ve bütünsel tarama arasında uygun bir denge kurulmasını sağlamakta ve GA ile ES algoritmalarının erken yakınsama problemini ortadan kaldırmaktadır. Arama parametrelerinin otomatik olarak belirlenmesi ve özuyarlanımı ile tasarımcı çabası en aza indirilmiştir. Önbesleme kısıtları için pürüzsüz ceza makanizmaları önerilmiş ve bahsedilen arama algoritmasına eklenmiştir. Yanıt yüzeyi modellemesi için, parçalı kübik Hermite eğriler kullanılarak verim-bilinçli sentez gerçekleştirilmiştir. SACSES'i sistem düzeyindeki sentez araçlarıyla birleştirmek için sıradüzenli bir sentez yapısı önerilmiştir. Sıradüzenli yapı, devre ve sentez düzeylerini birleştirmek için ek araçlara olan gereksinimi ortadan kaldırmaktadır. SACSES ve serim düzeyindeki sentez aracı, serim parazitlerinin devre başarımına olan etkilerinin en aza indirildiği bir geribesleme döngüsü ile birleştirilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

A_c	Common mode gain
A_d	Differential gain
$A_{v,in}$	Voltage gain due to the changes in input
$A_{v,ps}$	Voltage gain due to the changes in power supply voltage
g_{ds}	Output conductance
g_m	MOSFET transconductance due to gate to source voltage
g_{mbs}	MOSFET transconductance due to bulk to source voltage
h_{ij}	Hermite basis functions
$inrd$	Thermal noise current due to parasitic drain resistor
$inrs$	Thermal noise current due to parasitic source resistor
$J_g(v)$	Jacobian matrix of the function $g(v)$
k	Boltzmann's constant
$NLEV$	Noise model level parameter
r_{ds}	output resistance
T	Absolute temperature
Z_0	Characteristic impedance
Z_{in}	Input impedance
Z_L	Impedance of the load connected to port two of a two port network
Z_{out}	Output impedance
Z_S	Impedance of the load connected to port one of a two port network
α	Cooling rate
γ	Factor for the calculation of thermal noise of channel resistance
ϕ_i	Output noise component due to the i^{th} noise source
AC	Alternating Current
ALG	Analog layout generator

BTS	Basic two stage
CAD	Computer aided design
CMOS	Complementary metal oxide semiconductor
CMRR	Common mode rejection ratio
DA	Design automation
DC	Direct Current
GA	Genetic algorithms
GBW	Gain bandwidth product
EA	Evolutionary algorithms
EKV	Enz-Krummenacher-Vittoz
ES	Evolutionary strategies
FDCMFB	Fully differential common mode feedback
FOM	Figure of merit
HGA	Hierarchical genetic algorithm
IC	Integrated circuit
IGFET	Insulated-Gate Field Effect Transistor
LNA	Low noise amplifier
MC	Monte-Carlo
MNA	Modified Nodal Analysis
MOS	Metal oxide semiconductor
MOSFET	Metal oxide semiconductor field-effect transistor
NN	Neural network
NPOSA	New population-oriented simulated annealing
OPAMP	Operational amplifier
PDF	probability density function
PSO	Particle swarm optimization
PSRR	Power supply rejection ratio
Q	Quality Factor
SA	Simulated annealing
SACSES	Simulation-based analog circuit synthesizer using evolutionary strategies
SoC	System on chip

SPASE	Simulation program with analog synthesis emphasis
THD	Total harmonic distortion

1. INTRODUCTION

1.1. Motivation for Analog Design Automation

The invention of the transistor in 1947, together with the production of the first integrated circuit (IC) in 1958 led to a completely new lifestyle, which can be called *the electronic era*. In a few decades, electronic circuits replaced mechanical control mechanisms and new devices such as cellular phones, television and video equipment became a part of our lives. Today, one often thinks how people were able to live without the numerous electronic instruments around us. The ever-increasing complexity of electronic circuits led to a trend towards system on chip (SoC) designs, which are complete, mixed signal ICs containing millions of transistors.

When the sixty-year history of solid state electronics is inspected, it is noted that the key element is the continuous progress in terms of decreasing transistor size and increasing circuit complexity. The progress was so steady that Gordon Moore's prediction in 1965 [1] has been valid, more or less, throughout the last four decades. Also known as the Moore's Law, this observation states that the number of transistors integrated on a given silicon area will double every eighteen months. This exponential growth was possible because process engineers decreased the feature size continuously and designers were able to utilize this decrease by designing circuits with increased number of transistors. However, today's million transistor ICs and complicated system-on-chip designs necessitate increasing the engineer's productivity in order to cope with the advances in chip density [2].

An effective approach to increase productivity is to widen the scope of computer aided design (CAD) tools. Traditionally, CAD tools are mainly used as system analyzers or circuit simulators in simulate-update-re-simulate iterations. Such iterations take minutes although the simulation alone takes a few milliseconds. Most of the time

is consumed by the engineer's decision making, updating the system parameters and re-invoking the analyzer. If a search mechanism can be incorporated into the simulator, either rule-based or optimization-based, so that it can perform decision making, the whole design procedure can be accelerated several orders of magnitude. This conversion from an analyzer to a synthesizer is more complicated than it may seem because the resulting CAD tool has to give results competitive with at least those of an average engineer's in order to be useful. This poses the necessity of searching the whole design space and being robust. If simple search algorithms are used, the improvement in the iteration time is more than offset by the increase in the number of iterations.

For more than a decade, useful CAD tools for digital circuits have been available. Digital circuits are two-valued and hence they have finite input output combinations and are noise immune. Also, the hierarchical nature of digital circuits led to the establishment of the well-known design flow called the Y-chart. These facts resulted in the rapid development of digital CAD tools fitting well into the design methodology. Today, with the use of hardware description languages, it is possible to synthesize a digital circuit starting from verbal description with minimum human interruption. On the other hand, when designing an analog circuit, one needs to take into account the effects such as noise and coupling, choose the best among numerous topologies and work with nonlinear devices controlled by continuous free parameters. Analog circuits have less hierarchical nature, which makes it difficult to establish a general design flow. Consequently, analog CAD development has been much slower and analog CAD tools are years, if not decades, behind their digital counterparts. Hence, design of the analog section of a mixed signal application is performed manually by experts and even if the analog section has 10% of the total transistor count, it's design time dominates the overall design time. Also, these manual designs have higher possibility of possessing errors and are usually responsible for expensive design iterations [3].

Due to several reasons such as availability of automated design and test tools, ease of scaling and noise immunity, there is a trend to perform more circuit functions

digitally, and avoid time consuming analog designs. However, we live in a continuous valued world and whatever function the circuit performs, it must interface with the external world in an analog fashion. This means that input and output sections such as low noise amplifiers and output buffers as well as the sections interfacing the analog and digital parts of the circuit like analog-to-digital converters and sample-and-hold circuits will always remain analog [4]. Voltage, current and frequency references also require analog circuitry. In addition, some digital circuits such as microprocessors and data storage systems start to exhibit some analog characteristics as speed and density increases, and analog design techniques need to be utilized for their design. This indispensability of analog circuits increases the importance of analog CAD tools, making their development a very active research area.

1.2. Analog Design Flow

Analog design can be decomposed into three levels of abstraction, namely system, circuit and layout levels, as shown in Figure 1.1. The highest level, system level, is responsible for converting high level specifications to lower level block specifications. For example, if a bandpass filter is to be realized, the center frequency and bandwidth are input to the synthesizer. The tool selects the required order of the filter and optimizes the block specifications. In this case, the optimized block is usually an operational amplifier (OPAMP), and its power and area are to be minimized by changing its gain, bandwidth and output resistance while keeping the system specifications satisfied. In doing so, the system level tool requires some knowledge about the performance of the building blocks. Without this knowledge, the system level tool can converge to unfeasible solutions, or to solutions with high area and power values. This knowledge can be provided by a performance estimator, which is a model of the building block, or by communicating with the circuit level synthesizer.

Unlike circuit and layout levels, system level design tools are usually architecture-specific. At the system level, analog CAD tools have reached rather satisfactory results with systems characterized by regular structures such as switched

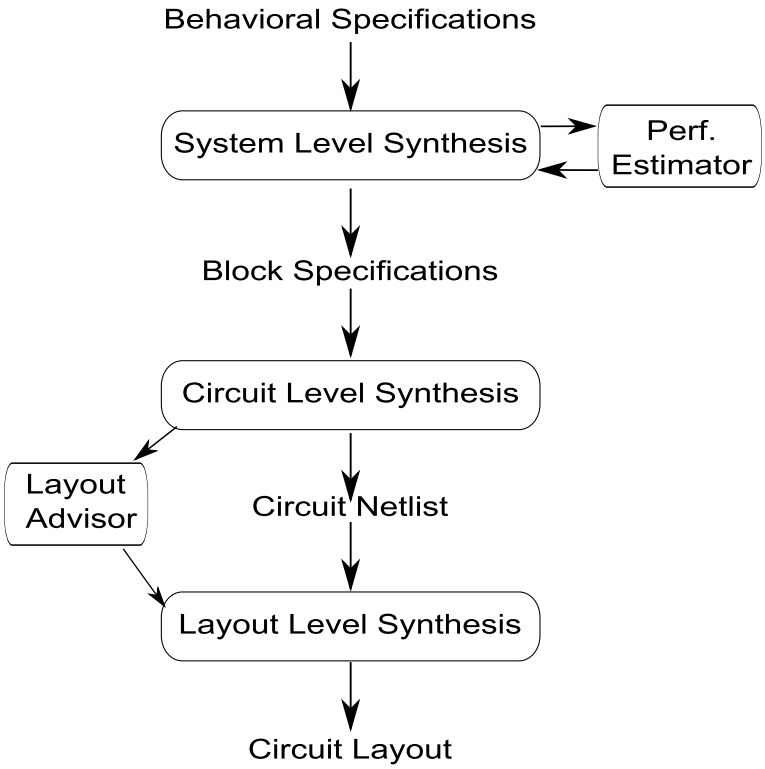


Figure 1.1. Analog design flow

capacitor filters [5, 6] and data converters [7]. Although these tools cover a substantial fraction of the analog circuits needed in most industrial applications, a more general approach, able to cope with arbitrary architectures is still needed.

Circuit level synthesis is at the center of the analog design flow and is responsible for transistor sizing and determination of resistor and capacitor values as well as bias voltages and currents. The synthesizer must be independent of topology and fabrication process; it must give accurate results and must consider process variations. One must differentiate between a circuit optimizer, which tries to improve a given feasibly sized circuit, and a circuit synthesizer, which starts with randomly sized circuits and searches the entire space defined by the variables in order to find the global optimum. The large, highly nonlinear search space with several local minima makes circuit level synthesis a difficult task.

The lowest level of analog design, layout level, is responsible for generating the chip layout based on the sizing information from the circuit level synthesizer. During layout generation, parasitic resistors, capacitors and inductors are inevitably formed. The layout generator should minimize these parasitics. However, trying to minimize all of the parasitics might be very complicated and a better method is to minimize the parasitics that affect performance metrics most. This performance-driven approach requires sensitivity analysis and/or performing iterations with the circuit level synthesizer. After generating the chip layout, the top-down analog design procedure is completed. The resulting layout is then verified with a bottom-up methodology starting from building block extraction, circuit and system simulation.

1.3. Objective of the Thesis

This thesis has two main objectives. The first one is to develop a simulation-based analog circuit synthesis methodology that requires minimum user effort and design knowledge. Methods for increasing circuit robustness in terms of insensitivity to device mismatches and process variations, and synthesizer robustness in terms of general applicability and automatic determination of search parameters will be provided. The second objective is to develop an infrastructure to communicate with the other levels of abstraction.

1.4. Key Features and Contributions

The analog synthesis methodologies presented in this dissertation have the following key features and contributions:

- The simulation-based analog synthesis tool, SACSES (Simulation-based Analog Circuit Synthesizer using Evolutionary Strategies), eliminates the accuracy problems and limitations of equation-based and model-based methods. Also, no equation or model generation time, which can be orders of magnitude longer than synthesis time, is required.
- In order to shorten the synthesis time and eliminate the drawbacks of using

a commercial simulator, an in-house written simulator that is fully synthesis oriented, SPASE (Simulation Program with Analog Synthesis Emphasis), is used. SPASE has several acceleration mechanisms and a robust structure that makes it suitable to be run in the inner loop of a search algorithm.

- Robustness of the synthesizer is increased by the use of several techniques:
  i) Biasing constraints are introduced by smooth penalty functions that allow transitions between different biasing schemes.
  ii) Metropolis criterion is embedded in the selection mechanism so as to minimize the risk of premature convergence.
  iii) With self-adaptation of strategy parameters and automatic calculation of annealing parameters, tuning the search parameters is not necessary.
  iv) Cost function weights are self-evolved with a mechanism to prevent cheating. To our knowledge, this is the first application of self-evolution of weights to circuit synthesis.
- Circuit robustness is increased with yield-aware synthesis. The presented yield estimation methodology is based on piecewise cubic Hermite models of performance metrics generated using simulation data. This is the first time Hermite splines are used for response surface modeling.
- A new synthesis methodology, hierarchical synthesis, that combines successive levels of abstraction, is developed together with the study of [8]. The method is applied to the system and circuit levels and the need for a performance estimator is eliminated.
- Circuit and layout levels of automation are linked with a feedback loop so that the effects of the layout parasitics are minimized by circuit sizing modifications.

1.5. Thesis Organization

The organization of the thesis is as follows:

Chapter 2 presents the classification of analog design automation methods and summarizes the previous studies related to the thesis.

Chapter 3 describes the methods and acceleration mechanisms used in the circuit simulator SPASE.

Chapter 4 gives the details of the ES-based search algorithm and related modifications. Techniques used for automation of search parameters are also explained. Circuit synthesis examples are given at the end of the chapter.

Chapter 5 is dedicated to the yield estimation methodology based on piecewise cubic Hermite response surface models.

Chapter 6 explains the hierarchical synthesis method and its application to combine system and circuit levels of automation.

Chapter 7 describes the feedback loop between the circuit and layout levels of automation.

Chapter 8 concludes the thesis.

2. BACKGROUND

Circuit level design automation can be classified into two main categories, namely knowledge-based and optimization based. Knowledge-based approach was the earlier approach in which designer expertise and design strategies for each topology were utilized in computer programs in terms of design equations and heuristics. Some examples are OASYS [9], which is a CMOS OPAMP compiler, BLADES [10], [11] and IDAC [12]. In order to derive a deterministic design flow, some simplifications are inevitable and simpler metal oxide semiconductor field-effect transistor (MOSFET) models are preferred in this approach. The resulting tools provide very fast synthesis but the results are usually sub-optimum and inaccurate. Another disadvantage of knowledge-based approaches is that they lack topology independence and require substantial amount of human effort to derive analytical equations and heuristics.

With the increase in computer performances, an alternative approach, optimization-based synthesis emerged, which transforms the problem into a function minimization problem and exploits search algorithms. Design knowledge can still be embedded in the search algorithm. For example, in [13], mathematical sizing rules for sub-blocks are generated in order to guarantee functionality and increase yield while in [14], a hierarchy of objective functions is used for adaptive adjustment of cost function weights.

During optimization, circuits can be evaluated by using either equations or simulation, which divides the optimization-based synthesis into two sub-categories. Some equation-based tools are AMGIE [3], OPASYN [15] and OPTIMAN [16]. AMGIE uses a symbolic analyzer to derive the small signal transfer function while the large signal equations are supplied by the user. Unlike most of the other tools, it can be used from system specifications to the layout. Such an approach of using analytical equations for circuit evaluation is attractive because it is faster and provides an insight to circuit

operation. However, due to the exponential growth in the number of terms in the transfer function with circuit complexity, simplification techniques are needed at the cost of degraded accuracy. Even the simplified, less accurate equations are too complex for human understanding. This drawback of the equation-based approach, together with the availability of high-speed computers resulted in a trend towards simulation-based approaches. This does not mean that equation-based approach is fully abandoned, and a recent example is [17], which also incorporates layout parasitics into symbolic equations. Also, in [18], an equation-based model is used for initial sizing of a two stage Miller OPAMP.

Simulation-based approach provides SPICE-level accuracy. Most of the simulation-based tools use commercially available, specific simulators. For example, the tool presented in [19] is integrated with the CADENCE Design Framework, GBOPCAD [20] uses HSPICE, [18] utilizes Eldo and its Levenberg-Marquardt algorithm-based optimizer. There are also some tools like MAELSTROM [21] and ANACONDA [22] that have simulator-encapsulating mechanisms in order to be more generic. By using a commercial simulator, the extra work of writing a simulator and possible programming errors are avoided. It is also easier to integrate the resulting tool with the corresponding commercial design environment. However, it has more significant disadvantages. In a synthesis tool, simulation time takes most of the total synthesis time, but most of the commercial simulators are developed for human operation and latencies due to license checks through a network or inter-application communication through text files are of little concern. Also, most commercial simulators are embedded in larger design environments that can crash and reset the synthesis process. In addition to avoiding these drawbacks, by writing a simulator, one takes full control over the optimization process. For example, in ASTRX/OBLX [23], a relaxed Kirchhoff's current law (KCL) is used at the earlier stages of optimization that results in fewer matrix solution iterations; and DELIGHT.SPICE [24] takes advantage of an enhanced SPICE simulator with the addition of DC (Direct Current), AC (Alternating Current) and transient sensitivity calculations. In this thesis, in-house developed accelerated DC, AC and noise simulators are used. The

DC simulator used in this thesis was implemented initially in [25] and was updated with Level-3 and BSIM3v3 MOSFET models in [26]. The acceleration mechanisms elaborated in Chapter 3 increase the simulation speed without sacrificing accuracy.

An intermediate between equation and simulation based methods have also emerged. This approach is model-based and uses circuit models prepared for each topology. Circuit evaluation using these models is faster than simulation and more accurate than simplified equations. Posynomial models have been developed for CMOS (Complementary Metal Oxide Semiconductor) OPAMPs [27, 28], multi-stage amplifiers [29], pipeline A/D converters [30] and oscillators [31]. The model is used in order to obtain a convex formulation for the optimization problem, which can be solved globally in a short time by using geometric programming. Since geometric programming requires all of the design objectives and constraints to be posynomial functions, simpler transistor models are used and some second-order effects are neglected. The time required for model development and topology dependence limit the usefulness of this approach. A similar tool is [32], which extracts a quadratic polynomial and posynomial performance model using both transistor level simulation data and results of statistical analysis of process and environmental variations. The model is valid for the feasible region around the given initial solution and is used for fine-tuning. Another example of the model-based approach is [33], where a two-layer neural network (NN) is used to model the target circuit's SPICE simulation data. The resulting model is used in a genetic algorithm based synthesizer. Synthesis takes a few seconds because of the faster NN model, while model development requires two hours. The model-based approach provides fast synthesis but model preparation is time consuming. Generating suitable training data to adequately represent the circuit's behavior is another problem. Also, the resulting model's accuracy may be unacceptable at certain regions of the search space.

Another part of a circuit synthesizer is the search algorithm. Classical local search algorithms like steepest descent are not adequate for synthesis purposes since

they can easily get stuck at one of the numerous local minima. Yet, their usage can be advantageous in terms of speed and problem formulation. For example, the tool given in [34] uses sequential quadratic programming (SQP) for performance space exploration and takes advantage of the fact that SQP yields solutions exactly on the constraint bounds. Classical search algorithms are more suitable for circuit optimizers, in which they perform fine-tuning of a readily feasible circuit. A synthesizer, on the other hand, starts from a randomly sized circuit and needs global search methods. Among various global search techniques, simulated annealing (SA) and evolutionary algorithms (EA) like genetic algorithms (GA) and evolutionary strategies (ES), have been widely used for the synthesis of analog circuits.

The simpler approach, simulated annealing, is a stochastic computational method and is analogous to the physical process of annealing. It converges to the global optimum provided that a stationary distribution is reached at each temperature and cooling is sufficiently slow. This guaranteed method can take a very long time and faster techniques such as very fast annealing [35] have been introduced. SA uses a single solution and compares it with an updated one at each step. In contrast, GA use a population of solutions and have parallel search capability. In GA, the possible variable combinations are encoded using genes composed of binary strings. A population of individuals containing different genes is evolved by recombination and mutation operators under the control of selection [36]. It turns out that diversity and focus, or global and local search, must be perfectly balanced with these three mechanisms [37]. Otherwise, the population fails to converge in the case of over-diversity or converges to a local minimum in the case of over-focus. In this regard, non-uniform mutation [36], which decreases the mutation step size as evolution proceeds, rank-based selection [36], which uses a ranking mechanism to rate the individuals before selection and Boltzmann selection [37], which uses a stochastic selection mechanism similar to that of simulated annealing, were introduced.

One drawback of GA which limits its usefulness for analog circuit synthesis is the increase in the number of representative bits for encoding the variables with required precision. In circuit sizing, biasing voltage and currents as well as capacitor and resistor values can be considered to be continuous. Transistor length and widths are discrete because of process limitations but their encoding still needs lengthy strings. In this regard, ES, which can be viewed as an extension of GA for continuous domains, are more suitable for analog circuit sizing. ES are implemented in the problem's own domain, preferably with the incorporation of domain specific knowledge. For continuous variables, ES provide a resolution limited only by the machine precision, not by the binary string length. Another advantage of ES is that incorporation of self-adaptation of strategy parameters such as mutation step size is easily performed in continuous domains [37].

Studies that contain improvements to the above algorithms, or even that suggest alternatives are also emerging. An algorithm called parallel recombinative simulated annealing [38], that combines ES and SA is used in MAELSTROM. M-DESIGN [39] uses differential evolution for the optimization of power amplifiers. In [40], a different version of DE, co-evolutionary DE, is proposed and applied to analog circuit optimization. In [41], particle swarm optimization (PSO), a population-based technique inspired by swarming of animals is used in RF circuit synthesis. The tool presented in [42] uses PSO algorithm for initial sizing of a self-biased complementary folded cascode and uses direct pattern search for optimizing the initial solution. GENOM [43] performs search space decomposition for improved parallel computing and increases the mutation rate when there is no evolution for a certain number of generations for premature convergence prevention. In [44], a Support Vector Machines (SVM) based learning scheme is added to GENOM for model generation. Another analog circuit optimizer, [19], uses least squares gradient search together with GA in order to reduce the total optimization time. In [45], an evolutionary algorithm based on SA is suggested. In this algorithm, called a new population-oriented simulated annealing (NPOSA), several SA nodes are run in parallel. Each node has its own temperature and updates it according to its rank in the population. By this way, each individual is able to raise its temperature

and escape from the local minimum if it is lagging behind the others. Alternatively, a solution that is much better than the others lowers its temperature for fine-tuning around the possible global optimum.

The search algorithm used in this thesis is a combination of ES and SA. The Boltzmann selection mechanism of SA is used in the selection step of ES. In this manner, the scope of the search is adjusted during synthesis. At the beginning, the population temperature is high and worse than average individuals have a chance to be selected for the next generation. Towards the end of synthesis, the temperature is low and search is limited to a narrow area around the possible global optimum. By adjusting the diversity and focus dynamically, this mechanism prevents premature convergence to a local minimum.

3. CIRCUIT SIMULATION

The simulator implemented in this thesis, SPASE, is capable of performing DC, AC and noise simulations. Additional routines utilizing the three simulators calculate several performance metrics such as gain, bandwidth, phase margin and noise figure. In order to increase the synthesis speed without sacrificing accuracy, these SPICE-like DC, AC and noise simulators have acceleration mechanisms suitable for circuit synthesis loops. Details of the simulators, acceleration mechanisms and additional routines are given in this chapter.

3.1. DC Simulation

The first stage of performance evaluation is DC simulation by which the operating point of the circuit is determined. AC and noise simulations rely on the linearized model around this operating point. Hence, high accuracy is critical in DC simulation. The model used for DC simulations is given in Figure 3.1. With the inclusion of parasitic source and drain resistances, the MOSFET becomes a six node device, and this increases the size of the linear system. It should be noted that Figure 3.1 is a linearized model. The task of the DC simulator is to determine a set of transconductance values g_m, g_{mbs}, and output conductance g_{ds} ($= 1/r_{ds}$) such that the resulting node voltages and branch currents obey the network equations.

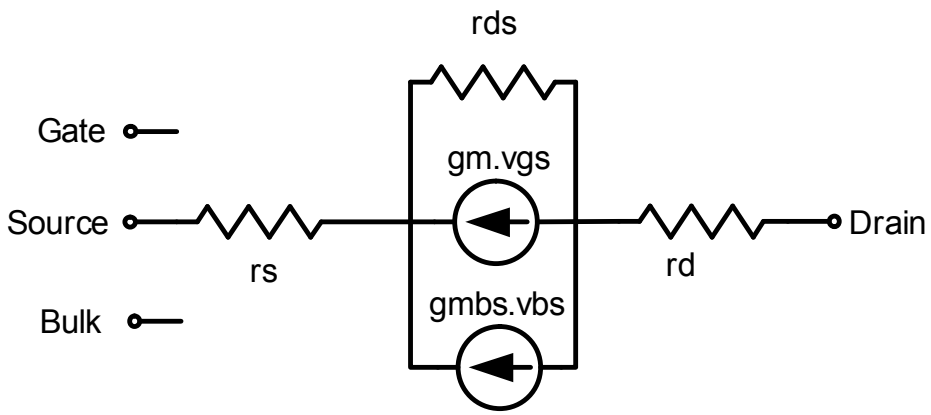


Figure 3.1. MOSFET model used for DC simulation

Due to several second order effects, MOSFETs exhibit complicated nonlinear characteristics and modeling MOSFET behavior has been an active research area. Several MOSFET models have been developed for use in the SPICE simulator. These models approximate the linearized MOSFET model parameters g_m, g_{mbs} and g_{ds} as multivariate functions:

$$g_m = f_1(v_{gs}, v_{ds}, v_{bs}, W, L) \tag{3.1}$$

$$g_{mb} = f_2(v_{gs}, v_{ds}, v_{bs}, W, L) \tag{3.2}$$

$$g_{ds} = f_3(v_{gs}, v_{ds}, v_{bs}, W, L) \tag{3.3}$$

where v_{gs} is the gate to source voltage, v_{ds} is the drain to source voltage, v_{bs} is the bulk to source voltage, W is the width and L is the length of the MOSFET. The simplest MOSFET model for the SPICE circuit simulator, the Level 1 model, often called Shichman-Hodges model [46], is a first order model and is suitable for long-channel transistors. Level 1 describes the current dependence on voltages for gate voltages greater than the threshold voltage; neglecting the sub-threshold current. Since its simplicity causes inaccurate results, use of Level 1 is limited to hand analysis. The Level 2 model addresses second-order effects associated with small-geometry devices. Unlike in Level 1, the sub-threshold current is not equal to zero. Level 2 is computationally very complex and convergence problems are often encountered [47]. Level 3 model is a semi-empirical model developed to address the shortcomings of Level 2. It runs faster than Level 2 and convergence problems are seldom encountered. However, the sub-threshold current and the output conductance are not properly modeled.

The Level 1, 2, and 3 models constitute the first generation of MOSFET models. With the rapid development of the MOS (Metal Oxide Semiconductor) technology in the 1980s, they became inadequate for simulating circuits with a large number of ever-

smaller transistors. Starting with BSIM (Berkeley Short-Channel IGFET Model) [48], a different modeling philosophy, which put less effort into developing physical models but instead concentrated on mathematics for faster and more robust circuit simulation [47], was adopted. Although BSIM was not suitable for use in analog designs because of negative output conductance problems, two later developments, BSIM2 and HSPICE Level 28 made second-generation simulators suitable for analog IC design [49]. Second generation models have an empirical and complex implementation and use several parameters without clear physical meanings.

BSIM3 and its extension BSIM4, along with MOS Model 9, began the third-generation approach which reintroduced the physical basis into the models. The use of smoothing functions in third-generation models provides continuous and smooth behavior of device characteristics across all the operating regions [50]. Third generation models also utilized a high number of parameters, most of which determined the characteristics of the smoothing functions. An analytical model, Enz-Krummenacher-Vittoz (EKV), requiring fewer parameters was also developed [51]. The EKV model is a fully symmetrical model which uses the bulk voltage as the reference.

The first version of the DC simulator [25] had only Level 1 and Level 2 support. This simulator was embedded in synthesis loops in [52] and [26]. During its search, a synthesizer visits circuits of very different characteristics and any convergence problem of the MOSFET model decreases the synthesizer's efficiency. Due to the convergence problems of Level 2, Level 3 support for the DC simulator was added in [26]. Although convergence problems were solved, due to inadequate modeling of output conductances in Level 3, output offset voltages and gains of operational amplifiers were not accurate. This accuracy problem was solved with the addition of BSIM3v3 support in [26]. BSIM3v3 is still widely used by the manufacturers and we did not need to add support for another MOSFET model during this thesis.

In this thesis, the AC and DC simulators both use the modified nodal analysis (MNA) formulation. For a circuit with n nodes and m voltage sources, the $(m+n) \times (m+n)$ MNA matrix has the following form [25]:

$$\begin{bmatrix} \mathbf{G} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{j} \end{bmatrix} = \begin{bmatrix} \mathbf{i} \\ \mathbf{e} \end{bmatrix} \tag{3.4}$$

Each matrix element is explained in the following:

- **G:** $(n \times n)$ admittance matrix determined by the interconnections of the passive circuit elements and voltage controlled current sources. Each entity in the diagonal is equal to the sum of the conductances of the elements connected to the corresponding node. For example, the third diagonal element is the sum of the conductances connected to node 3. The off diagonal elements are negated conductances of the elements connected to the pair of corresponding node. For a resistor R between nodes j and j', *-1/R* goes into the **G** matrix at locations (j,j') and (j',j). For a voltage controlled current source connected between nodes k and k' with a controlling voltage between nodes j and j', the negated transconductance *-g* goes into the locations (k,j') and (k',j). If an element is grounded, it will only have contribute to one entry in the G matrix, at the appropriate location on the diagonal. If it is not grounded it will contribute to four entries in the matrix, two diagonal entries and two off-diagonal entries.

- **B:** $(n \times m)$ matrix composed of 0, 1 or -1's, determined by the connection of the voltage sources. Each location in the matrix corresponds to a particular voltage source (first dimension) or a node (second dimension). If the positive terminal of the i^{th} voltage source is connected to node k, then the element (k,i) in the **B** matrix is a 1. If the negative terminal of the i^{th} voltage source is connected to node k, then the element (k,i) in the **B** matrix is a -1. If no voltage source is connected, the entity is zero.

- **C:** $(m \times n)$ matrix composed of 0, 1,-1's or voltage gain values, determined by the connection of the voltage sources. Similar to the **B** matrix, if the positive

terminal of the i^{th} voltage source is connected to node k, then the element (i,k) in the **C** matrix is a 1. If the negative terminal of the i^{th} voltage source is connected to node k, then the element (i,k) in the **B** matrix is a -1. If no voltage source is connected, the entity is zero. If the voltage source is controlled by the voltage between nodes j and j', the voltage gain, μ is inserted into location (i,j') and $-\mu$ is inserted into location (i,j). The **B** and **C** matrices are closely related, particularly when there are only independent voltage sources. In this case, **C** is the transpose of **B**.

- **D:** $(m \times n)$ matrix determined by the connection of the voltage sources. D is zero unless the circuit has current controlled voltage sources.

- **v:** $(n \times 1)$ matrix of unknown voltages.

- **j:** $(m \times 1)$ matrix of unknown currents through the voltage sources.

- **i:** $(n \times 1)$ matrix containing the sum of the currents through the passive elements into the corresponding node. Each element is either zero, or equal to the sum of independent current sources to that node.

- **e:** $(m \times 1)$ matrix of the values of the independent voltage sources.

Due to the aforementioned nonlinear characteristics of MOSFETs, one-step solution of the network equations is not possible. Instead, an initial set of node voltages is assumed and linearized MOSFET parameters are calculated from (3.1), (3.2), and (3.3), according to the SPICE model. Then, Newton-Raphson iterations are used to converge to the operating point. For a set of nonlinear equations, $g(v) = 0$, this method suggests the following iterations:

$$v^{(i+1)} = v^i - J_g^{-1}(v^i)g(v^i) \tag{3.5}$$

where $J_g(v)$ is the Jacobian matrix of the function $g(v)$:

$$J_g(v) = \begin{bmatrix} \frac{\partial g_1}{\partial v_1} & \frac{\partial g_1}{\partial v_2} & \cdots & \frac{\partial g_1}{\partial v_n} \\ \frac{\partial g_2}{\partial v_1} & \frac{\partial g_2}{\partial v_2} & \cdots & \frac{\partial g_2}{\partial v_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial v_1} & \frac{\partial g_n}{\partial v_2} & \cdots & \frac{\partial g_n}{\partial v_n} \end{bmatrix} \tag{3.6}$$

In nonlinear DC analysis, the Jacobian consists of the nodal conductance matrix of the linear elements together with the linearized conductances of each nonlinear element [25]. In order to avoid matrix inversion, for which computational cost is high, both sides are multiplied by $J_g(v^i)$ and the following linear system of (3.7) is obtained. This reduces the long operation count from n^3 to $n^3/3$.

$$J_g(v^i)v^{i+1} = J_g(v^i)v^i - g(v^i) \tag{3.7}$$

With the new set of node voltages, v^{i+1}, new MOSFET parameters are calculated, the Jacobian matrix is re-computed and (3.7) is solved again. Iterations are continued until the difference between successive solutions is below a given tolerance or a maximum number of iterations is reached.

SACSES utilizes two mechanisms to increase the DC simulation speed. First, since the locations of the non-zero elements of the Jacobian matrix are fixed throughout the iterations, sparse matrix techniques are employed in order to decrease the number of multiplications. Another possibility for accelerating the simulator is the use of pre-determined starting points. Unlike standard SPICE, which uses a starting point consisting of zero node voltages, SPASE uses the node voltages taken from the previous solution as the starting point when called again. This increases simulation speed if the current circuit is close to the previous one. So, a search algorithm that slightly perturbs the circuit sizing at each iteration, such as simulated annealing, better utilizes the accelerated simulator. Genetic and ES-based algorithms also take advantage of this

acceleration mechanism, especially at the later stages of evolution, when the population focuses on possible global minimum. This situation is shown in Figure 3.2 where the average number of iterations is plotted for each generation with and without the acceleration mechanism. When the acceleration mechanism is applied, fixed initial node voltages are only used for the first individual of the initial population. For the other individuals, node voltages of the previously simulated individual are used as the starting point. Although initially the population is composed of circuits with very different transistor sizes, it is clear that the mechanism is useful even at this stage. The reason is that the circuits have a common topology and the voltage distribution of one circuit resembles that of another. Hence, it is better to start with the previous circuit's node voltages rather than a fixed set of voltages. It is observed that the acceleration rate improves as the population converges and the overall average number of iterations during the whole search is 18.3 without, and 8.3 with the mechanism.

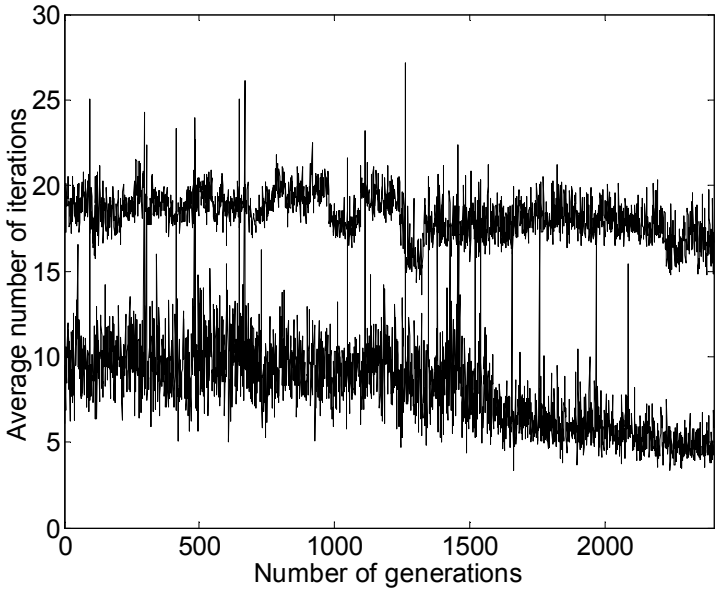


Figure 3.2. Average number of DC iterations for each generation with (bottom) and without (top) the acceleration mechanism

3.2. AC Simulation

Once the operating point and linearized MOSFET parameters are accurately calculated, some of the AC performance metrics like gain and bandwidth can be calculated

using equations with acceptable accuracy. Correspondingly, in our previous studies, we have used methods other than direct simulation to evaluate AC performance. In [52], user-given equations and neuro-fuzzy models were used, both of which were highly topology dependent. In order to decrease the effort required when migrating to a new circuit topology, a tool that solves the given simplified modified nodal analysis (MNA) matrix was written and used in [26]. Although this method was easier to apply to a new topology, the simulator still lacked the ability to calculate performance metrics such as unity gain frequency and phase margin accurately. This accuracy problem is more pronounced for high order topologies. In order to overcome this accuracy problem and increase the synthesizer's topology independence, a generic AC simulator was implemented in this thesis.

The AC simulator reads the circuit netlist, forms the MNA matrix using the linearized MOSFET model parameters obtained from DC simulation, and solves the resulting linear system. The linearized MOSFET model of Figure 3.3 is used to generate the MNA matrix. Usually, there is only one AC source and the matrix size is $(n+1) \times (n+1)$. For a circuit with several biasing sources, this means that the matrix size of AC simulation is smaller than that of DC simulation. In addition, no Newtonian iterations are necessary since the small signal model of Figure 3.3 is composed of only linear elements. However, unlike DC simulation, for which the MNA matrix is real valued, the MNA matrix for AC simulation is complex valued. Hence, solution of a complex valued linear system is required for AC analysis, and an LU decomposer supporting complex numbers was written to solve the linear system. Additional operations used for complex-valued arithmetics slow down AC simulation, and on the average an AC simulation takes as much time as a DC simulation. On the other hand, the current version of the AC simulator does not take advantage of the sparsity of MNA matrix, meaning that there exists a margin for improvement of AC simulation times.

Using AC simulation, DC gain, bandwidth, unity gain frequency and phase margin can be evaluated. Except DC gain, obtaining the response for a single frequency

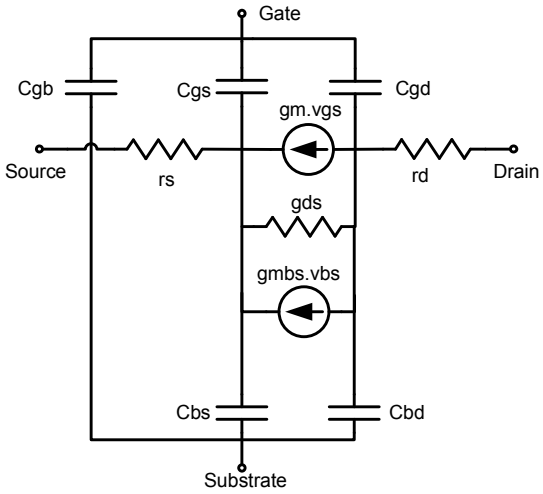


Figure 3.3. Mosfet model used for AC simulations

is not enough in order to evaluate the above metrics, and the AC response curve has to be traced. This search is performed by a tracer routine that uses Brent's method [53]. In order to decrease the number of necessary AC simulations, the routine starts from the results of the previous circuit. For a two stage OPAMP circuit, six AC simulations are required on the average for determining the bandwidth with 1% tolerance.

Evaluation of common mode rejection ratio (CMRR), power supply rejection ratio (PSRR), input impedance (Z_{in}), output impedance (Z_{out}) and s-parameters using an AC simulator is also possible. Dedicated routines for each of these performance metrics that alter the circuit configuration and re-run the simulator have been implemented. Common mode rejection ratio, CMRR, of a differential circuit is defined as the ratio of differential and common-mode voltage gains of a circuit:

$$CMRR = \frac{A_d}{A_c} \tag{3.8}$$

where A_d is the differential gain, and A_c is the common mode gain of the circuit. Previously, CMRR was calculated by user given approximate equations for each topology. However, a simple manipulation of the current simulators allows the calculation of an exact value. Differential voltage gain, A_d, is already calculated using the AC simulator. Common-mode gain is calculated by a routine that connects the supply to both of the

input nodes and re-runs the AC simulator. CMRR is frequency dependent and usually deteriorates with frequency. This behavior can be observed by running the routine for different frequencies. This is an important advantage over equation-based methods, as the frequency dependence characteristics of CMRR are harder to approximate. Similar to the CMRR, power supply rejection ratio, PSRR, of a circuit is defined as:

$$PSRR = \frac{A_{V,in}}{A_{v,ps}} \tag{3.9}$$

where $A_{v,in}$ and $A_{v,ps}$ denote the voltage gains due to the changes in input and power supply voltage, respectively. PSRR gives an idea on how much circuit performance deviates from ideal due to power supply ripple. $A_{v,in}$ is already available from AC simulation. A routine is written to connect a signal source to supplies and re-run the AC simulation to find $A_{v,ps}$. Then, PSRR is calculated from 3.9. For circuits having symmetrical supplies, PSRR is usually different for each supply and this is taken into account by calculating $A_{v,ps}$ for each supply. Frequency dependence of PSRR can also be calculated using the same routine.

The z-parameters and s-parameters are especially needed to synthesize radio frequency (RF) circuits such as low noise amplifiers (LNAs). For the two-port network of Figure 3.4, z-parameters are defined by (3.10) [54], and can be calculated from (3.11). After the solution of the linear system of (3.4), both node voltages v and currents through the voltage sources j are available. The z-parameter routine connects a voltage source to the input or output node, successively, and by measuring the current through the source and the other node's voltage, calculates the z-parameters according to the (3.11).

$$\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix} \begin{pmatrix} I_1 \\ I_2 \end{pmatrix} \tag{3.10}$$

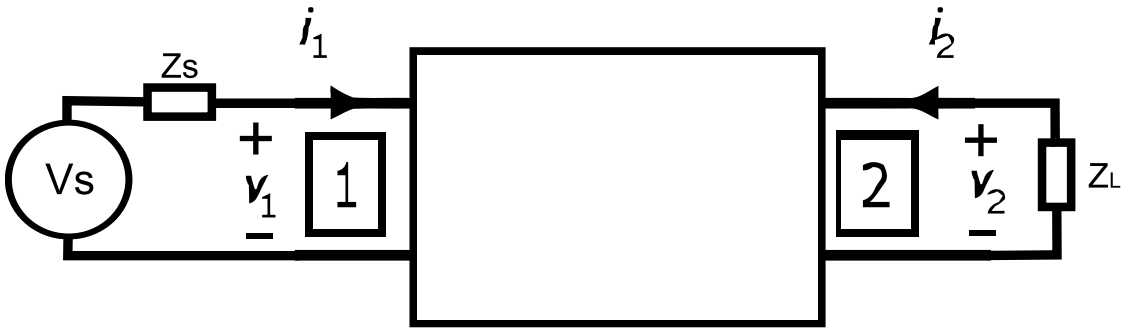


Figure 3.4. Voltage and current conventions for a two-port network

$$\begin{aligned} Z_{11} &= \tfrac{V_1}{I_1}|_{I_2=0} \quad Z_{12} = \tfrac{V_1}{I_2}|_{I_1=0} \\ Z_{21} &= \tfrac{V_2}{I_1}|_{I_2=0} \quad Z_{22} = \tfrac{V_2}{I_2}|_{I_1=0} \end{aligned} \tag{3.11}$$

The input impedance, Z_{in}, and the output impedance Z_{out} can be calculated from:

$$Z_{in} = z_{11} - \tfrac{z_{12}z_{21}}{z_{22}+Z_L} \qquad Z_{out} = z_{22} - \tfrac{z_{12}z_{21}}{z_{11}+Z_S} \tag{3.12}$$

where Z_S and Z_L are the impedances of the loads connected to ports one and two, respectively. Once the z-parameters are obtained, s-parameters can be easily calculated using conversion formulas 3.13 [54] for the given characteristic impedance, Z_0.

$$\begin{aligned} s_{11} &= \tfrac{(Z_{11}-Z_0)(Z_{22}+Z_0)-Z_{12}Z_{21}}{\Delta} \qquad s_{12} = \tfrac{2Z_0Z_{12}}{\Delta} \\ s_{21} &= \tfrac{2Z_0Z_{21}}{\Delta} \qquad\qquad\qquad\qquad s_{22} = \tfrac{(Z_{11}+Z_0)(Z_{22}-Z_0)-Z_{12}Z_{21}}{\Delta} \end{aligned} \tag{3.13}$$

where

$$\Delta = (Z_{11} + Z_0)(Z_{22} + Z_0) - Z_{12}Z_{21} \tag{3.14}$$

3.3. Noise Simulation

Noise performance is becoming more important with today's low power, small-area-per-transistor analog circuits. For a given topology, the designer can use his/her a priori knowledge to determine the dominant noise sources and write approximate noise

equations. However, because of its global search ability, the synthesizer must be able to evaluate a topology without any sizing assumptions. Even if the resulting circuit satisfies the conventional sizing assumptions, the intermediate circuits, especially those at the earlier stages of synthesis, are quite different and they should be evaluated correctly. Hence, such approximate equations, which also limit topology independence, are not suitable for a circuit synthesizer and the use of a noise simulator is necessary. In a MOSFET circuit, all resistors, including parasitic drain, source and channel resistances of MOSFETs, exhibit thermal noise. Thermal noise can be modeled as a current source connected between the terminals of the corresponding resistor. For a bandwidth of Δf, the mean square thermal noise of a resistor is:

$$\bar{I}^2_{n,th} = 4kT\Delta f/R \tag{3.15}$$

where k is the Boltzmann's constant, T is the absolute temperature in degrees Kelvin and R is the resistance value. Thermal noise for the parasitic drain and source resistances can also be calculated using (3.15). Channel resistances of MOSFETs also exhibit thermal resistances but for nonzero V_{DS} values, the channel resistor of a MOSFET is not uniform and the associated thermal noise should be calculated by integration along the whole channel [55], which results in:

$$\bar{I}^2_{n,th} = 4kT\gamma g_m \tag{3.16}$$

where the factor γ depends on the transistor parameters and bias conditions, and g_m is the MOSFET transconductance. SPICE uses $\gamma = 2/3$, which is the value at the saturation point and is a good approximation for long channel devices. However, higher γ values are necessary for channel lengths below 1 μm. Today, with the advance of manufacturing processes, most technologies have channel lengths far below 1 μm and the hard-coded value for γ is becoming a more pronounced handicap for SPICE. In fact, γ can be as high as 2.5 for some 0.25 μm devices [56]. To overcome this problem, designers add parallel current sources to increase the effective value of gamma and to approximate the results given by the noise parameter sheets of the manufacturer. In SPASE, this problem is eliminated by letting the designer input the value of γ.

In addition to channel thermal noise, MOSFETs exhibit flicker noise due to the random trapping of charge at the oxide-silicon interface of MOSFETs [56]. Owing to its inverse proportionality to frequency, flicker noise is also known as $1/f$ noise. Although its effect seems negligible for today's high frequency circuits, nonlinearities in mixers or oscillators can easily translate the low-pass spectrum to the radio frequency (RF) range. Flicker noise is usually represented as a voltage source in series with the gate:

$$\bar{V}_{n,fl}^2 = \frac{K}{WLC_{ox}} \frac{1}{f} \tag{3.17}$$

where K is a process-dependent constant. For simulation purposes, however, a current source representation is more convenient. This can be achieved by multiplying the noise voltage by MOSFET transconductance, g_m, and connecting the resulting current source, given by (3.17), between the drain and source terminals of the transistor:

$$\bar{I}_{n,fl}^2 = \frac{K}{WLC_{ox}} \frac{g_m}{f} \tag{3.18}$$

More accurate flicker noise models have also been developed [57], and are embedded in SPASE. Instead of a single parameter, K, these models employ two parameters, KF and AF. Together with the MOSFET noise model level parameter, $NLEV$, flicker noise is calculated from:

$$\bar{I}_{n,fl}^2 = \frac{1}{f} \frac{KF \cdot I_{ds}^{AF}}{COX \cdot L^2} \ , \ NLEV = 0 \tag{3.19}$$

$$\bar{I}_{n,fl}^2 = \frac{1}{f} \frac{KF \cdot I_{ds}^{AF}}{COX \cdot W \cdot L} \ , \ NLEV = 1 \tag{3.20}$$

$$\bar{I}_{n,fl}^2 = \frac{1}{f^{AF}} \frac{KF \cdot g_m^2}{COX \cdot W \cdot L} \ , \ NLEV = 2 \tag{3.21}$$

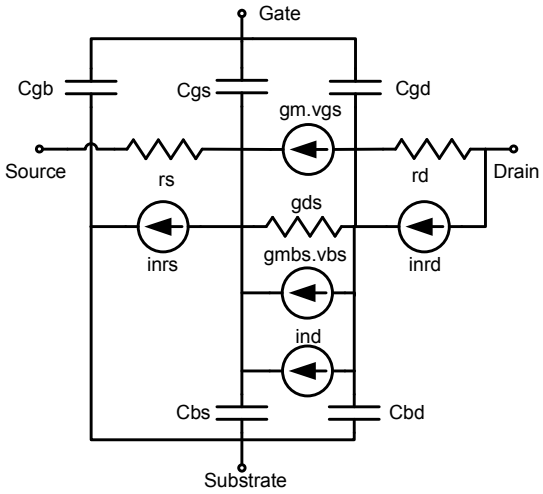


Figure 3.5. Mosfet model used for noise simulations

Figure 3.5 shows the MOSFET model used in the noise simulator, where *inrd* and *inrs* denote the thermal noise currents due to parasitic drain and source resistances. Since channel and flicker noises are in parallel, their mean square values are combined to give *ind*, assuming uncorrelated sources. After the noise sources are modeled, the contribution of each source to the output noise should be calculated. A simple approach to this problem is to connect each source one by one and run successive AC simulations. However, this approach requires as many simulation runs as the number of noise sources, n_{noise}:

$$n_{\text{noise}} = n_{\text{resistor}} + 3n_{\text{mosfet}} \tag{3.22}$$

where n_{resistor} is the number of resistors and n_{mosfet} is the number of MOSFETs. A neater approach is to use the adjoint vector formulation [58], which requires runtime of only one AC simulation to calculate output noise for a given frequency. The formation and LU decomposition of the MNA matrix is performed only once, and the contribution of each source is calculated by simple back-substitutions. The adjoint vector formulation starts with writing the MNA equation without the input voltage source, which gives the simplified form of:

$$\mathbf{YV} = \mathbf{I} \tag{3.23}$$

For each noise source, a separate $\boldsymbol{I}$ vector should be formed. In this sense, noise simulation problem can be viewed to be the solution of the above linear system for different right hand sides:

$$\mathbf{Y}\mathbf{V}_i = \mathbf{I}_i \tag{3.24}$$

$$\phi_i = \mathbf{d}^{\mathbf{t}}\mathbf{V}_i \tag{3.25}$$

where ϕ_i is the output noise component due to the i^{th} noise source and $\boldsymbol{d}$ is a constant vector of 0, -1 and 1's defining the relationship between the desired output and node voltages. For example, if the output of a differential amplifier is defined as the voltage difference between the nodes l and m, then the l^{th} element of the $\boldsymbol{d}$ vector is 1, m^{th} element is -1, and the remaining elements are 0. Solving (3.24) for $\boldsymbol{V}_i$ and substituting in (3.25) gives:

$$\phi_i = \mathbf{d}^{\mathbf{t}}\mathbf{Y}^{-\mathbf{1}}\mathbf{I}_{\mathbf{i}} = -\left(-\mathbf{d}^{\mathbf{t}}\mathbf{Y}^{-\mathbf{1}}\right)\mathbf{I}_{\mathbf{i}} \tag{3.26}$$

Note that the term in parentheses in 3.27 is constant for every noise source. The adjoint vector is defined to be equal to the transpose of this term:

$$\mathbf{X}^a = \left(-\mathbf{d}^{\mathbf{t}}\mathbf{Y}^{-1}\right)^{\mathbf{t}} \tag{3.27}$$

Instead of evaluating the inverse of Y matrix, the adjoint vector is calculated by solving the equation (3.28) using LU decomposition and back substitution. Then, the output for each noise source is calculated from (3.29)

$$\mathbf{Y}^{\mathbf{t}}\mathbf{X}^{\mathbf{a}} = -\mathbf{d} \tag{3.28}$$

$$\phi_i = -\left(\mathbf{X}^a\right)^{\mathbf{t}}\mathbf{I}_i \tag{3.29}$$

It should be noted that only one LU decomposition is required for the evaluation of output noise; the rest is the application of (3.29), which is simply a vector multiplication. Similar to SPICE, the noise sources are assumed to be uncorrelated, and the overall output noise can be calculated by using the contributions of each noise source:

$$\phi = \sqrt{\phi_1^2 + \phi_2^2 + \cdots + \phi_{nnoise}^2} \tag{3.30}$$

Noise performance is usually specified in terms of noise figure, *NF*. Noise figure is a measure of how the signal to noise ratio, *SNR*, is degraded by a device and is expressed as:

$$NF = 10 \log \left(\frac{SNR_{in}}{SNR_{out}} \right) \tag{3.31}$$

where SNR_{in} and SNR_{out} are the signal to noise ratios at the input and output, respectively. Noise figure is calculated by a routine that runs the simulator for successive frequencies and calculates the signal to noise ratios.

3.4. Comparison of SPASE and HSPICE

Figure 3.6 compares the results of SPASE and HSPICE for the AC analysis of a BTS (Basic Two Stage) OPAMP. Note that the almost identical results give overlapped plots of gain and phase. The good agreement in AC simulation also implies good agreement in DC simulation since AC simulation depend on the linearized MOSFET models obtained from DC simulation. In order to better illustrate the accuracy of SPASE, two performance metrics of two different BTS OPAMPs, obtained from SPASE and HSPICE simulations, are compared in Table 3.1. The only noticeable difference is in the output offset voltage, V_{os}, row. In a BTS OPAMP, this node is very sensitive to transistor parameters and internal parasitic resistances since there is no common mode feedback.

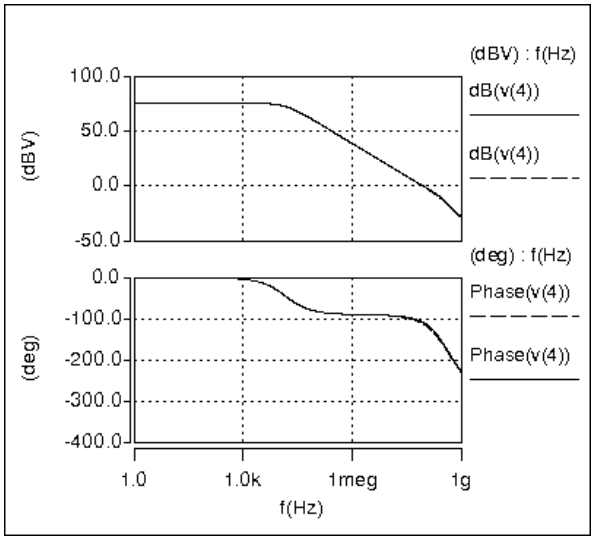


Figure 3.6. AC response of a BTS OPAMP calculated with SPASE (dashed lines) and HSPICE (solid lines). The plots are overlapped due to very close results of SPASE and HSPICE

Table 3.1. Performance metrics obtained from SPASE and HSPICE

Circuit	**BTS-1**			**BTS-2**		
Specification	**SPASE**	**HSPICE**	**Error (%)**	**SPASE**	**HSPICE**	**Error (%)**
A_0 (dB)	70.42	70.43	0.01	78.28	78.30	0.03
BW (kHz)	25.87	25.38	1.9	5.54	5.64	1.8
r_{out} (kΩ)	16.52	16.49	0.2	4.327	4.332	0.1
V_{os} (mV)	38.13	35.81	6.5	64.79	66.49	2.6
Power (mW)	1.587	1.587	0	4.125	4.122	0.07
PM	73°	72°	1.4	25°	26°	3.8
CMRR (dB)	102.5	103.3	0.8	99.3	99.6	0.3

4. CIRCUIT SYNTHESIS

The circuit synthesis flow utilized in this work is shown in Figure 4.1. The first step is to read the circuit topology given in SPICE netlist format. Circuit specifications are embedded as comments in the same netlist. The netlist can be generated manually by the designer or automatically by a system level synthesizer like [59]. The next step is to read the search parameters, such as population size or initial temperature, and a list of variables to be optimized. The list of variables includes range, stepsize and continuity information for each variable together with inter-variable relationships. The inter-variable relationships, such as variables having a fixed ratio, are mainly used to introduce matched transistors to the synthesizer. By manipulating the list, the designer can fix a certain variable or narrow its range. The final step before entering the search loop is the generation of initial solutions. Since a synthesizer must find a solution regardless of the starting point, generation of randomly sized circuits is adequate for this step.

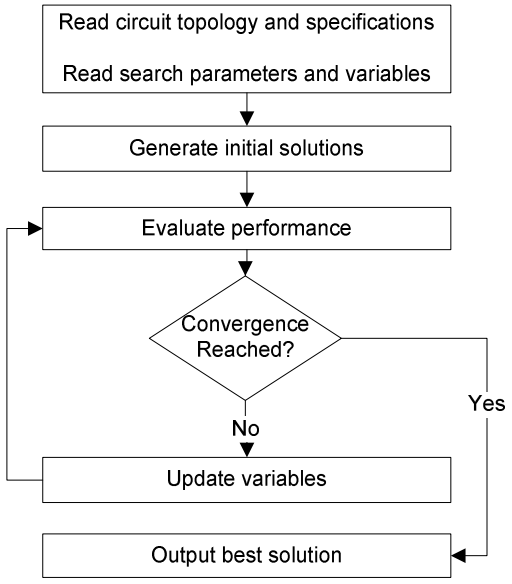


Figure 4.1. The proposed circuit synthesis flow

The search loop is composed of performance evaluation, variable updating and deciding whether the search should be ended or not. The decision is based on checking

if maximum number of generations is reached or if the population is frozen. The latter condition uses the proposed strategy parameter of population temperature, which will be explained in this chapter. The decision step consumes negligible time compared to the other two steps, so the time required for performance evaluation together with the synthesis algorithm's overhead determines the speed of the synthesis tool. Unlike evaluation of analytic multivariate functions, simulation of a circuit requires iterative solution of a nonlinear system and consumes substantial amount of CPU time, usually much longer than the overhead of the algorithm. This means that a complex, high overhead optimization algorithm can be used without noticeable decrease in synthesis time. On the other hand, the simulators must be as fast as possible without sacrificing any accuracy, since circuit synthesis is the final stage before layout generation and fabrication. Hence accelerated simulators having SPICE accuracy are utilized in this study, and are accompanied with a robust ES algorithm that has global search capability.

This chapter starts with the description of the cost function. After explaining the general structure of the ES algorithm, details of the recombination and mutation operators as well as selection and annealing mechanisms are given. Mechanisms for automated detection or self-adaptation of search parameters are also described. The chapter ends with the presentation of synthesis examples for different circuits.

4.1. Cost Function

The cost function combines the simulation results and enables the transformation of the constrained circuit design problem into an unconstrained function minimization problem. In doing so, it must not introduce extra complexities while handling the constraints of the design problem. Since cost function is the only means for the search algorithm to judge the circuit's performance, an inappropriate cost function may cause the algorithm to get stuck at local minima. In this study, the cost function is composed of two parts, one being performance related, $C_{perf.}$, and the other being constraint

related, $C_{penalty}$:

$$C = C_{perf.} + w_{penalty} C_{penalty} \tag{4.1}$$

where $w_{penalty}$ is the weight for the biasing constraints. By adding a penalty term based on the violation of constraints to the cost function, the constrained problem is transformed to an unconstrained one. The performance related part, $C_{perf.}$, is the weighted sum of squared normalized distances from the target point:

$$C_{perf.} = \sum_{i=1}^{n} w_i \hat{f}_i^2 \tag{4.2}$$

where, n is the number of performance specifications and the normalized values $\hat{f}_i$ are calculated from (4.3) using good, g_i, and bad, b_i, limits given by the user. The minimum limit of zero on the normalized value is necessary to allow over-satisfaction of some performance criteria without increasing $C_{perf.}$.

$$\hat{f}_i = \frac{g_i - f_i}{g_i - b_i}, \quad \hat{f}_{i,\min} = 0 \tag{4.3}$$

Normalization allows the weights to represent relative importance of each specification, independent of the specific ranges of performance metrics. The good and bad limits constitute the only required design knowledge about the given topology. If the designer is unfamiliar with the topology, he/she can set high values for g_i and search the topology's limits. In case of this over-estimation of topology's capabilities, SACSES tries to reach the given good values, and the topology's limits for each performance criteria can be explored. As explained in the previous chapter, while some performance metrics are directly available after simulation, some performance metrics are calculated using dedicated routines. SPASE is able to calculate a wide range of performance metrics, such as gain, bandwidth, phase margin, input and output impedances, DC and AC CMRR and PSRR, s-parameters, noise figure, etc. The designer can also introduce equation-based performance metrics based on simulator output. For example, slew

rate, SR, can be calculated using the output current and capacitance. All of these metrics are normalized and included in the cost function of (4.2)

In addition to the performance specifications, biasing constraints should be included in the cost. In analog circuits, MOSFETs are usually biased in the saturation region except the ones that are used as resistors, such as those in a frequency compensation circuit. A circuit with transistors biased in the triode region may satisfy the specifications according to the AC analysis results but if we run a transient analysis to verify the circuit, we most probably see that the circuit's performance deteriorates with increasing input amplitude. Such circuits are also highly prone to performance degradations due to process variations [13]. Since transient analysis and the corresponding post-processing is computational-wise and storage-wise too costly to embed in the search iterations, the transistors are constrained to operate in the user given regions. If a transistor is out of the specified region, a penalty term should be added to the cost function that does not introduce any discontinuities or highly steep regions. Otherwise, the search algorithm will converge to the first constraint-satisfying region regardless of its performance. One approach that satisfies this need is to generate a penalty term for each transistor proportional to its distance from the constraint-satisfying point:

$$p_{cut-off} = \sum_{i=1}^{m} p_{cut_i} = \sum_{i=1}^{m} \frac{k_{th} V_{th_i} - V_{gs_i}}{V_{th_i}}, \quad p_{cut_i,min} = 0 \tag{4.4}$$

$$p_{triode} = \sum_{i=1}^{m} p_{triode_i} = \sum_{i=1}^{m} \frac{k_{sat} V_{dsat_i} - V_{ds_i}}{V_{dsat_i}}, \quad p_{triode_i,min} = 0 \tag{4.5}$$

$$C_{penalty} = p_{cut-off} + p_{triode} \tag{4.6}$$

where m is the number of biasing constraints and the factors k_{th} and k_{sat} are utilized to introduce a safety margin for biasing. Usually, it is better to let the synthesizer decide the amount of gate overdrive voltage; so, k_{th} should be close to unity. However,

higher values can be used if the synthesizer fails to bias the circuits properly, or if the designer wants to decrease noise sensitivity with increased gate overdrive. In the synthesis examples presented in this chapter, both k_{th} and k_{sat} are equal to unity. Inspecting 4.4, 4.5 and 4.6, we note that for circuits with all transistors in the user-specified regions, $C_{penalty} = 0$ and it increases continuously as the transistors get out of the corresponding regions. Thus, no discontinuity is added to the search space. This smooth increase in the cost value guides the search algorithm to properly bias the circuit. It also allows transitions from one properly biased region to another and prevents premature convergence to the first properly biased region found.

4.2. Search Algorithm

A $(\mu+\lambda)$ ES algorithm is used in this study with a modified selection mechanism. At generation, $(g+1)$, the population P_{μ}^{g+1} is formed by selecting μ individuals among the μ individuals of the previous generation and their λ offspring. The outline of the algorithm is given in Figure 4.2. Each individual's chromosome is composed of circuit variables $\boldsymbol{x}$, transistor widths, lengths, resistor, capacitor, inductor, biasing current and voltage values, and strategy parameter set $\boldsymbol{s}$:

$$\mathbf{x} = \{W_{1...J}, L_{1...J}, R_{1...K}, C_{1...M}, l_{1...N}, I_{1...P}, V_{1...Q}\} \tag{4.7}$$

$$\mathbf{s} = \{\sigma_{1...\mu+\lambda}, a_{1...\mu+\lambda}, w_{1...\mu+\lambda}\} \tag{4.8}$$

In (4.7), W_i denotes transistor widths, L_i denotes transistor lengths, R_i denotes resistors, C_i denotes capacitors, l_i denotes inductors, I_i denotes biasing currents and V_i denotes biasing voltages. In (4.8), σ_i denotes mutation step sizes, a_i denotes recombination coefficients, and w_i denotes cost function weights for each individual. The mutation stepsizes, crossover coefficients and cost function weights are included in the chromosome, so that their values also evolve as the search proceeds. By this way, the need for optimizing the strategy parameters is eliminated.

$g \leftarrow 0$

$P_\mu \leftarrow P_{\mu_0}$

while *convergence not reached* **do**

for $i = 1$ to $\lambda/2$ **do**

$[I_{parent1} \quad I_{parent2}] \leftarrow choose(P_\mu, 2)$

$[I_{\mu+i} \quad I_{\mu+i+1}]_s \leftarrow recombine_s(I_{parent1}, I_{parent2})$

$[I_{\mu+i} \quad I_{\mu+i+1}]_x \leftarrow recombine_x(I_{parent1}, I_{parent2})$

end for

for $i = 1$ to $\mu + \lambda$ **do**

if I_i *is selected for mutation* **then**

$I_{i_s} \leftarrow mutate_s(I_i)$

$I_{i_x} \leftarrow mutate_x(I_i)$

end if

$I_{i_{cost}} \leftarrow evaluate(I_i)$

end for

$P_\mu^{g+1} \leftarrow select(P_{\mu+\lambda}^g, \mu)$

$g \leftarrow g + 1$

$T \leftarrow update\_temperature()$

end while

output $\leftarrow$ *best solution*

Figure 4.2. Pseudocode of the ES algorithm

4.2.1. Initialization

The initialization process is composed of generating μ individuals with search variables randomly generated according to the given variable ranges. In circuit synthesis, some variables like transistor widths and lengths can have only discrete values because of the limitations of the manufacturing technology. It is possible to treat these variables as continuous variables and round the optimized values to the nearest practically possible value at the end of synthesis. However, due to the nonlinear characteristics of the large signal behavior of analog circuits, this approach causes a considerable deviation from the target performance values. Our approach is to initialize these variables with

discrete values and use practical values throughout the whole optimization so that no rounding is necessary at the end of the synthesis.

4.2.2. Recombination

In recombination, two parents are randomly chosen and two offspring are formed by crossing over their strategy parameters and search variables using the recombination coefficient a:

$$\begin{aligned} \mathbf{s}_{child_1} &= a \cdot \mathbf{s}_{parent_1} + (1-a) \cdot \mathbf{s}_{parent_1} \\ \mathbf{s}_{child_2} &= (1-a) \cdot \mathbf{s}_{parent_2} + a \cdot \mathbf{s}_{parent_2} \\ \mathbf{x}_{child_1} &= a \cdot \mathbf{x}_{parent_1} + (1-a) \cdot \mathbf{x}_{parent_1} \\ \mathbf{x}_{child_2} &= (1-a) \cdot \mathbf{x}_{parent_2} + a \cdot \mathbf{x}_{parent_2} \end{aligned} \tag{4.9}$$

First, the recombination coefficients are evaluated and the rest of (4.9) is applied with the new a values. This type of crossover is usually called arithmetical crossover [37]. The recombination coefficient,a is in the range $[0, 1]$. When $a = 0.5$, as in [52], the two offspring are identical and have averaged variable values. Averaging may seem logical but the behavior of analog integrated circuits is quite different from living beings. For example, a circuit with averaged variable values of two well-performing circuits may not even satisfy the biasing conditions. This possibility is especially high for some integrated implementations in which biasing and circuit function is not decoupled. An example is a BTS OPAMP, of which first stage biases the next one in addition to providing some gain. It was observed during this study that setting $a = 0.7 \ldots 0.9$ is a better choice since this gives offspring closer to one of the parents and this prevents the generation of constraint-violating individuals. The current version of SACSES allows self-adaptation of recombination coefficients and it is observed that a_i values converge to 0.9 or 0.1, although initially all $a_i = 0.5$.

4.2.3. Mutation

In the mutation procedure used by SACSES, each individual, I_i, has its own standard deviation for each search variable x_{ij}, recombination coefficient a_i, and cost function weights w_{ik}. During mutation, if an individual I_i, is selected for mutation, these standard deviations are first updated according to the equation:

$$\begin{aligned} \sigma_{ij}^{'} &= \sigma_{ij}\ e^{(\tau_0 N(0,1)+\tau N_i(0,1))} \\ \sigma_{a_i}^{'} &= \sigma_{a_i}\ e^{(\tau_0 N(0,1)+\tau N_i(0,1))} \\ \sigma_{ik}^{'} &= \sigma_{ik}\ e^{(\tau_0 N(0,1)+\tau N_i(0,1))} \end{aligned} \tag{4.10}$$

where $N(0,1)$ is a normally distributed random variable with expectation zero and standard deviation one, and $N_i(0,1)$ is a normally distributed variable that is sampled anew for each search variable. The recommended values for the parameters τ_o and τ are [37]:

$$\tau_0 = \frac{1}{\sqrt{2n}} \quad , \quad \tau = \frac{1}{\sqrt{2\sqrt{n}}} \tag{4.11}$$

where n is the dimension of the search space. After updating the standard deviations,σ_{ij}, search variables x_{ij}, recombination coefficient a_i, and cost function weights w_{ij} are updated using the equation:

$$\begin{aligned} x_{ij}^{'} &= x_{ij} + \sigma_{ij}^{'}\ N_i(0,1) \\ a_i^{'} &= a_i + \sigma_{a_i}^{'}\ N_i(0,1) \\ w_{ik}^{'} &= w_{ik} + \sigma_{ik}^{'}\ N_i(0,1) \end{aligned} \tag{4.12}$$

4.2.4. Selection

One major drawback of ES-based search algorithms and genetic algorithms is that they may get stuck at a local minimum if an inappropriate selection mechanism is used. In [60] , incorporating the Metropolis criterion of simulated annealing into

the selection mechanism was proposed. Some versions of this technique are known as Boltzmann selection [37] and used in parallel recombinative simulated annealing [38]. Metropolis criterion is used to choose between the current solution I_i and an alternative solution I_j in which I_i wins with logistic probability:

$$p(I_i) = 1/e^{(C_i - C_j)/T} \tag{4.13}$$

where C_i is the cost of I_i. In [60], offspring compete with their parents, a method which only makes use of local information. In this study, Metropolis criterion is used in a different manner. When selecting μ individuals among $(\mu + \lambda)$, a randomly chosen individual competes with a pseudo-individual having the average cost C_{av} and wins with the probability:

$$p(I_i) = 1/e^{(C_i - C_{av})/T} \tag{4.14}$$

and the process is repeated until μ individuals win against the average cost. By this way, global performance information is utilized. The presence of very low performance individuals increases the average cost and raises the acceptance probability to close to unity. This results in random selection of individuals even at low temperatures. In order to prevent this influence, the outliers should be neglected during the calculation of average cost, preferably with minimum introduction of elitist nature. Our approach is to neglect an individual I_i in the average cost calculation if $C_i > kC_{min}$, where C_{min} is the minimum cost in the population and k is a constant determining the neglection threshold. In this study, $k = 100$ is used. It was observed that lower values tend to increase the elitist nature of selection and cause premature convergence while higher values are ineffective. Although very small, the outliers still have a chance of survival since they are also subjected to (4.14).

By incorporating the Metropolis criterion into the selection mechanism, a new control parameter, namely, population temperature, is introduced. At higher temperatures, it is highly probable to choose a bad individual and this contributes to population

diversity. As the temperature is lowered, the population focuses on a certain region of the search space, fine-tuning the variables and this near-end stage is rather a local search around the supposed global optimum. In this regard, population temperature can be viewed as a measure for the maturity of the evolution process, with lower temperature meaning later stages of evolution.

The proposed selection mechanism and elitist selection are compared in Figure 4.3, where the evolution of the normalized costs with the two algorithms is plotted. Since the cost is normalized, weights for all of the performance metrics are equal to unity. Both searches start with a best individual cost of around 100 converge after 4200 generations, 134400 circuit evaluations. Although elitist selection quickly decreases the cost of the best individual, it causes the search algorithm to get stuck at a local minimum with a cost of 0.02. On the other hand, the use of Metropolis criterion slows down the initial stage of evolution but helps the search algorithm reach a solution having a much lower cost of 0.0014. Resulting circuits both satisfy the performance constraints but the Metropolis-based algorithm is able to further decrease the area from 26500 m2 to 23000 m2 and power from 2.8mW to 2.2mW.

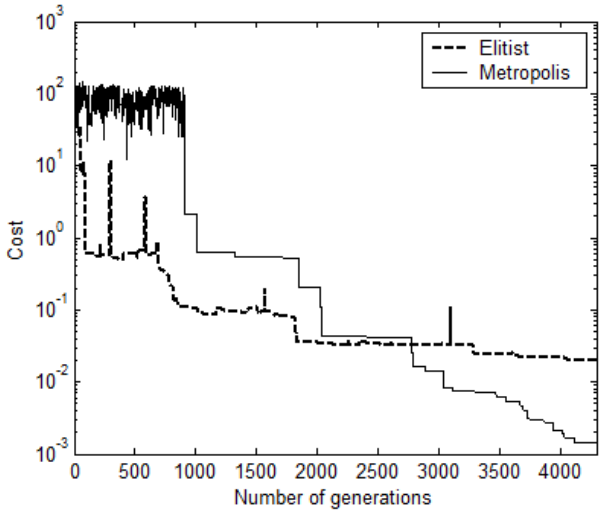


Figure 4.3. Evolution of the cost of the best individual with elitist and Metropolis criterion-based proposed selection mechanisms

4.2.5. Annealing Schedule

Temperature updating needs four new parameters, initial and final temperatures, T_0 and T_f, cooling rate α and Markov chain length T_{repeat}. A powerful heuristic is to choose the initial temperature such that 80% of worse than average individuals are selected by (4.14) [61]. For the final temperature, this rate must be only a few percent. The corresponding temperature values are problem specific but can be determined before starting the synthesis by running several iterations and taking the average acceptance percentages. If T_0 and T_f are not specified by the user, SACSES starts with T_0 =1000 and calculates the average probability of choosing worse than average individuals, p_{worse}, by performing 100 trials of (4.14). Then, T_0 is updated using the corrector equation:

$$T_0^{n+1} = T_0^n + T_0^n(0.8 - p_{worse})/p_{worse} \tag{4.15}$$

With the updated value of T_0, SACSES performs 100 trials of (4.14) again and continues the iterations until $0.8 < p_{worse} < 0.85$. Usually, 4-6 iterations are enough to find an appropriate initial temperature. Once T_0 is determined, SACSES performs a similar procedure to find T_f, starting from $T_f = 1$ and setting target $P_{worse} = 0.01$.

The standard cooling schedule, Boltzmann annealing, is given by:

$$T_k = T_0/ln(k) \tag{4.16}$$

where k is the number of the annealing step. Ideally, one should check whether a Boltzmann distribution, and hence thermal equilibrium, is reached before updating temperature. However, the time required to finish such a cooling process is not necessarily finite and exponential cooling schemes such as [62] and [35] were proposed. SACSES uses a similar cooling scheme, with limited Markov chain length:

$$T_k = \alpha\, T_{k-1} \tag{4.17}$$

Markov chain length is limited by the parameter T_{repeat}, which determines the number of generations that must pass before updating the temperature. SACSES starts annealing with $T_{repeat} = 1$ and increments T_{repeat} for every decade decrease in temperature. This is parallel to the fact that the Markov chain length required to reach thermal equilibrium grows with decreasing temperature [63]. Using the values for T_0, T_f and α, total number of generations can be calculated from:

$$N_g = \left[\frac{\left(\log\left(T_0/T_f\right)\right)\left(\log\left(T_0/T_f\right)+1\right)}{2} \right] \left[\frac{\log\left(T_f/T_0\right)}{\log\alpha} \right] \tag{4.18}$$

where the value in left brackets is the average number of generations before temperature is updated, and the value in right brackets is the number of temperature updates. Cooling rate α, is calculated by imposing a constraint on the minimum number of generations, $N_{g_{min}}$ before completing the synthesis. An appropriate choice, obtained from our experiments, is $N_{g_{min}} = 2000$. Since T_0 and T_f are known, cooling rate can be calculated from (4.18) . It should be mentioned that the calculated value of T_f is not used as a strict stopping criterion and evolution is continued until the population fails to improve for successive 100 generations. Hence, the actual number of generations is usually larger than 2000.

4.2.6. Self-adaptation of Weights

In [52], only mutation step sizes were self-adapted. In this study, recombination coefficients and weights are also self-adapted. Self-adaptation of weights were problematic due to the algorithm's tendency to "cheat", i.e. to equate the weights to zero to minimize the cost function. A method to overcome this problem has been suggested by Eiben et. al. [64]. Their suggestion is to use tournament selection between two individuals and use the maximum of the individuals' weights for evaluation. The selection mechanism used in SACSES is different. SACSES utilizes Metropolis selection on maximum weights of all individuals. When evaluating individuals, overall maximum weights of the population are used to prevent the algorithm from cheating. By this way, the previously described selection method of modified Boltzmann selection, which

prevents premature convergence and allows fine-tuning at the later stages, is kept. Also, unlike tournament selection, all of the individuals are evaluated using the same set of weights and so, their rank in the population can be clearly identified. This information has several uses. For example, knowing the best individual is necessary for recording the algorithm's state and calculating dynamic range of the population is necessary for complex annealing schedules and convergence detection methods.

Initially all of the weights are equal to unity. As the algorithm proceeds, crossover and mutation operators allow them span a large range of values. When one of the performance metrics cannot be satisfied for a high number of generations, it is observed that the corresponding weights increase. Figure 4.4 shows the evolution of area and weight of $area$, w_{area}, used in the cost function of a BTS OPAMP. Acceptable limit was set at $area < 30000\mu m^2$. The self-adaptation mechanism can be clearly observed. Initially, high area values decrease with the increase in weight. Then, very low weight values cause a rise in total area around generation 1000. In this region, other performance metrics are optimized. Next rise in weight values, near generation 1500, decrease the area to acceptable levels. After generation 2500, when other performance constraints are satisfied, weight is further increased to minimize total area. It should be noted that this seemingly complex weight adaptation is performed through self-evolution of weights, without any user intervention. Another run with the weights equal to unity were also performed. In this run, the synthesizer was not able to decrease the output offset voltage below the acceptance threshold even after 12000 genrations. After increasing the corresponding weight and re-starting the evolution, convergence was reached after 9000 generations. These observations show that, compared to pre-determined set of weights, self-adapted weights provide higher robustness and convergence speed. Also, this approach does not require any user effort to optimize the weights.

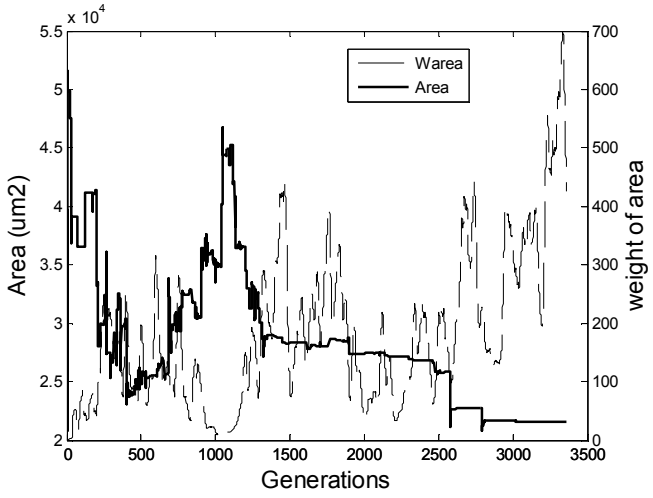


Figure 4.4. Evolution of area and w_{area} for a BTS OPAMP

4.3. Circuit Synthesis Examples

The performance of the proposed circuit synthesizer has been tested by designing several circuits, five of which are given in this section. Target technology is a standard 0.35 μm CMOS technology with 3.3 V supply voltage. The first circuit is a BTS OPAMP, a commonly used topology for testing design automation tools. The second one is a more complex topology, a fully differential OPAMP with common mode feedback (FDCMFB). The third circuit is a Folded Cascode (FC) OPAMP, selected to demonstrate the synthesis tool's performance in folded architectures. The fourth circuit is a low noise amplifier (LNA), and this example illustrates the capabilities of SACSES for RF design. The final circuit is a High Swing Cascode Differential Current Integrator (HSCDCI), and in this example, results of hand-design and automated design are compared.

In all of the examples, a (30+20) ES scheme was used.The given synthesis durations are the averaged values of five runs with maximum deviations given in parentheses. Synthesis runs are performed on a single core 2.4 GHz PC. The search algorithm described in this chapter is easily parallelizable. If a network of processors is available, runtimes can be shortened almost linearly by assigning the evaluation of each individual to a different node.

4.3.1. BTS OPAMP

The circuit schematic of the synthesized BTS OPAMP is given in Figure 4.5. Since this is an IC implementation, the first stage biases the second stage in addition to providing some gain. Also, there is no common-mode feedback to adjust the output offset voltage. These limitations make the design of a BTS OPAMP with low output offset voltage, high gain and low output impedance more difficult than it may seem. Table 4.1 shows the target and attained performance values with a 1 pF load capacitor. The number of independent variables is 22, and convergence is reached after 3000($\pm$500) generations, in 18($\pm$3) minutes. Evaluation of 50 individuals takes 350 ms while the search algorithm overhead is only 6 ms, verifying the arguement at the beginning of this chapter that the algorithm overhead is negligible when compared to simulation time.

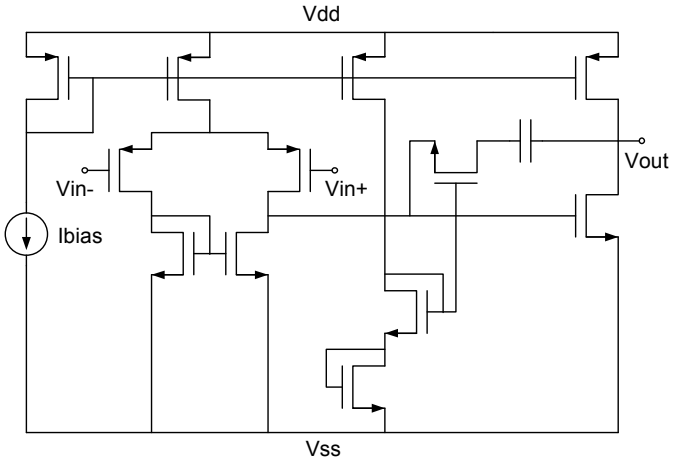


Figure 4.5. BTS OPAMP circuit schematic

Studies with synthesis examples of a similar BTS circuit exist. In [40], synthesis of a Miller-compensated BTS opamp with 14 variables takes 166 minutes, again on a 2.4 GHz PC. In [22], synthesis result of a BTS opamp with 18 variables is given. Synthesis duration is 168 minutes using a pool of 16 300 MHz SUN Ultra10 and 4 dual 300 MHz Ultra2 workstations. The reason for different number of variables is

Table 4.1. Specifications of the synthesized BTS OPAMP

Specification	**Target**	**Synthesis**	**HSPICE**
A_0 (dB)	> 75	75.5	75.5
BW (kHz)	> 10	14.4	14.4
r_{out} (kΩ)	< 50	28.5	28.5
V_{os} (mV)	< 10	2.8	2.3
PM	$> 60°$	69°	71°
CMRR (dB)	> 80	94	94
SR (V/μs)	> 50	60.9	58.6
Power (mW)	< 5	1.9	1.9
Area (μm^2)	< 30000	21600	-

the differences in the biasing and compensation circuits. Although direct comparison of runtimes is not possible due the use of different programming languages and/or computation platforms, it is clear that the combination of SACSES and SPASE can synthesize a similar circuit in a competitive time.

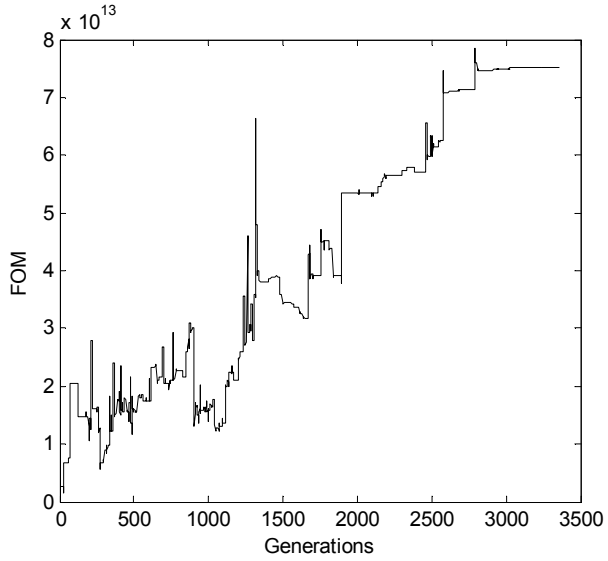


Figure 4.6. Improvement of figure of merit during synthesis

Figure 4.6 shows the improvement of the figure of merit (FOM) of $G \cdot BW/(Power \cdot Area \cdot r_{out})$ during a typical synthesis run. It should be noted that all of the specifications mentioned in Table 4.1 are included in the cost function but only four of them are combined to get a simpler FOM plot. During the final generations, the tool improves phase margin, which results in a slight decrease in FOM. Note that during the earlier evolution phase, when population temperature is high, down-moves are more frequent. Figure 4.7 shows the trade off between gain and bandwidth. The synthesizer quickly converges to the Pareto-optimal front and trades some gain for increased bandwidth.

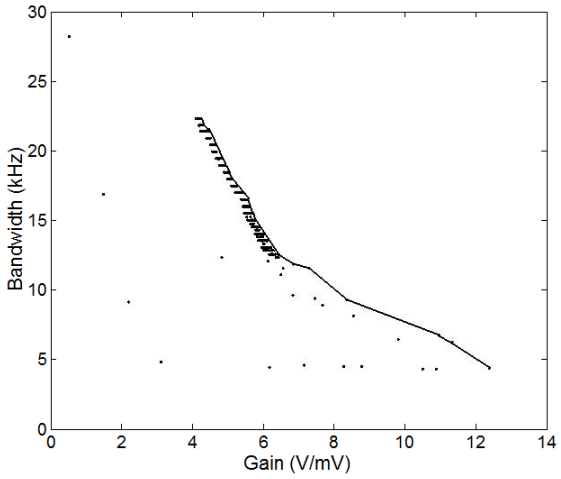


Figure 4.7. Gain-BW trade-off explored during synthesis. Solid line shows the Pareto-optimal front

4.3.2. FDCMFB OPAMP

The circuit schematic of the synthesized FDCMFB OPAMP [65] is given in Figure 4.8. Transistors M1-M14 constitute the fully differential amplifier while transistors M58-M64 provide common mode feedback to the input stage of the amplifier. The original circuit of [65] did not have the resistors in series to the compensation capacitors C_c. SACSES was not able to increase the phase margin beyond 35° with that circuit. Hand design trials confirmed the original topology's limited phase margin capability. Since all three stages of the amplifier have negative gain, nested Miller compensation is not possible and the circuit of [65] is modified with the addition of pole-zero compensation in order to improve phase margin. After this modification, SACSES is able to

synthesize circuits with phase margins in excess of 60°. Table 4.2 shows the target and attained performance values. Convergence is reached after 3500(±500) generations, in 25(±4) minutes.

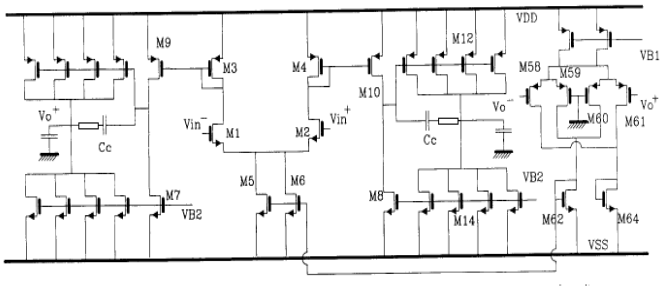


Figure 4.8. FDCMFB OPAMP circuit schematic

Table 4.2. Specifications of the synthesized FDCMFB OPAMP

Specification	**Target**	**Synthesis**	**HSPICE**
A_0 (dB)	> 75	79.3	79.3
BW (kHz)	> 5	6.5	6.5
r_{out} (kΩ)	< 10	4.1	4.1
V_{os} (mV)	< 10	8.7	8.2
PM	$> 60°$	65°	64°
Power (mW)	< 10	6.9	6.9
Area (μm^2)	< 100000	53800	-

Figure 4.9 shows the transient response of the synthesized OPAMP in open-loop configuration for an input of 1 kHz, 150 μV_{peak} signal. The output is rail-to-rail with less than 1% total harmonic distortion (THD), as shown in Figure 4.10. This shows that, with the help of biasing constraints described in the beginning of this chapter, the transistors are biased properly in the saturation region.

4.3.3. FC OPAMP

The circuit schematic of the synthesized FC OPAMP is given in Figure 4.11. Convergence is reached after 3200(±500) generations, in 21(±3) minutes. Table 4.3

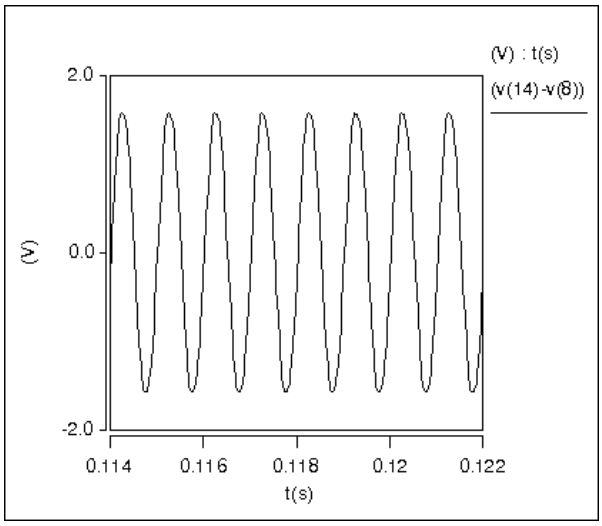


Figure 4.9. Transient response of the synthesized FDCMFB OPAMP

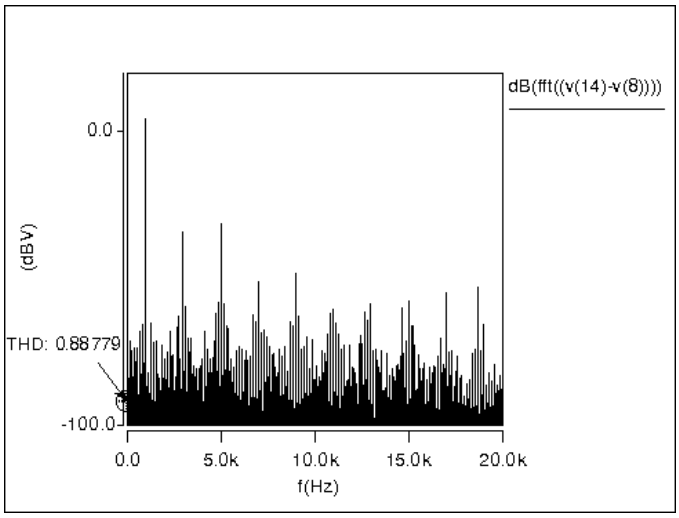


Figure 4.10. FFT analysis of the transient response of FDCMFB OPAMP

shows the target and attained performance values.

4.3.4. LNA

The circuit schematic of the synthesized LNA is given in Figure 4.12. Since this is an integrated implementation, on chip inductors are used, which can be modeled using parasitic inductor and capacitors as in Figure 4.13. 50 different inductors with inductance values ranging from 1 nH to 30 nH, and quality factor (Q) values ranging from 1.8 to 8.7 at 900 MHz are available from the target 0.35 μm CMOS technology. Their inductance values together with parasitic resistance and capacitance values are

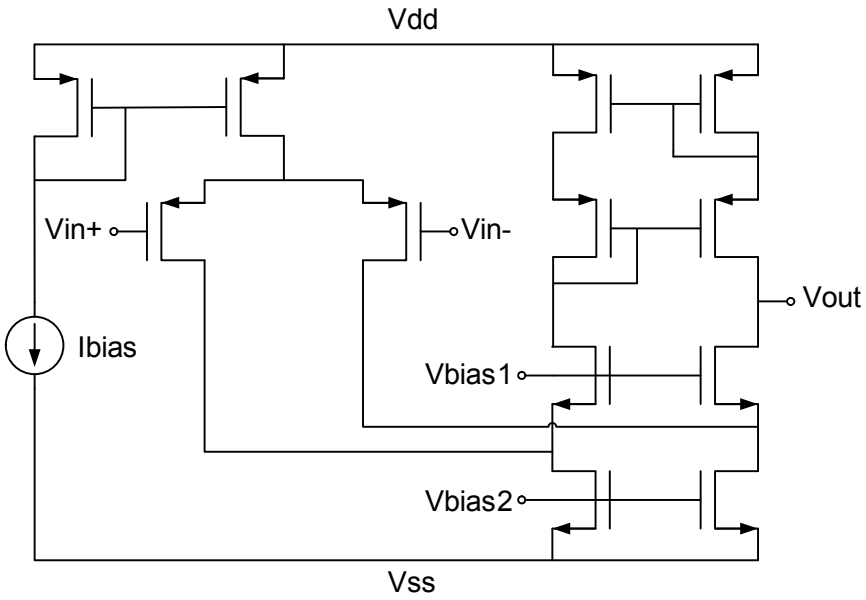


Figure 4.11. FC OPAMP circuit schematic

combined in a list from which the synthesizer selects.

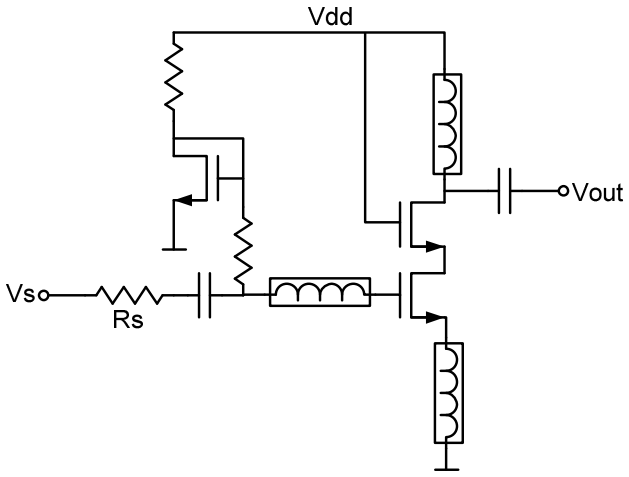


Figure 4.12. LNA circuit schematic

Table 4.4 shows the target and attained performance values. Convergence is reached after 5000(±1000) generations, in 15(±3) minutes. Similar to the other synthesis examples, the supply voltage is 3.3 V, and the input and output impedances are matched to 50 Ω. Gain and noise figure values are calculated at the target frequency of 900 MHz.

Table 4.3. Specifications of the synthesized FC OPAMP

Specification	**Target**	**Synthesis**	**HSPICE**
A_0 (dB)	> 75	75.8	75.8
BW (kHz)	> 10	16	16
V_{os} (mV)	< 100	68	70
PM	$> 60°$	62.4°	62.1°
CMRR (dB)	> 65	69	69
SR (V/μs)	> 10	12.2	12.0
Power (mW)	< 10	7.1	7.1
Area (μm^2)	< 10000	8430	-

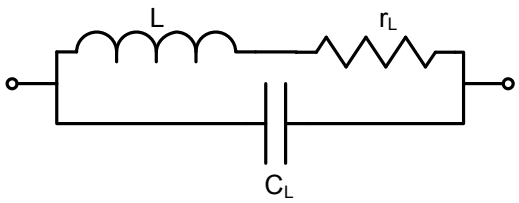


Figure 4.13. Model for on chip inductors

4.3.5. HSCDCI

The circuit schematic of the HSCDCI is shown in Figure 4.14 . The DC simulator was used to determine offset voltage and power dissipation. AC performance metrics were calculated using equations, which give more accurate results for this circuit than the BTS OPAMP. Proper synthesis of this integrator also necessitates calculation of THD. Exact measurement of THD requires a long transient analysis followed by a Fast Fourier Transform (FFT) analysis of the resulting waveform, both of which were unavailable. Also, this method consumes several orders of magnitude more time and is not suitable to be called at every iteration. So, a simpler but accurate enough method was developed which uses only one additional DC simulation per iteration. First, the voltage deviation Vc across the capacitors is calculated using the formula:

$$V_c = \frac{I_{in}}{C} \int_0^{T/2} sin(2\pi ft)dt = \frac{I_{in}}{\pi fC} \tag{4.19}$$

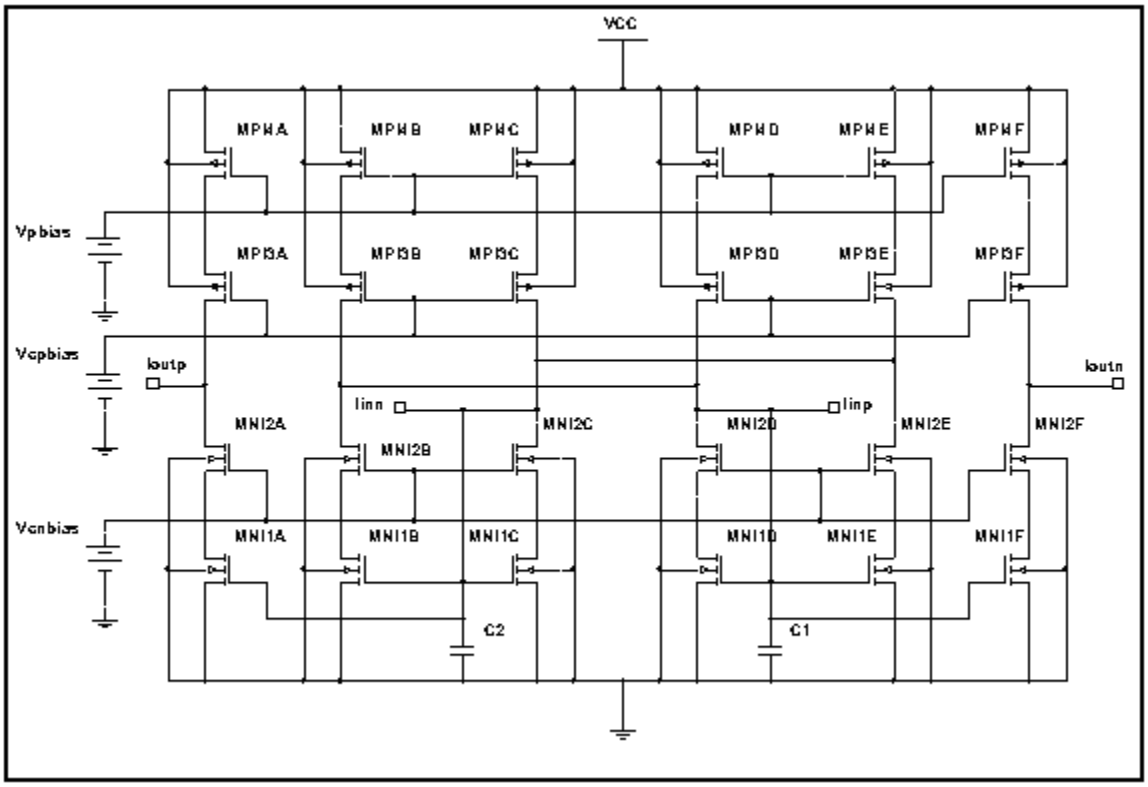


Figure 4.14. HSCDCI circuit schematic

This deviation changes the g_m of the output transistor, which is the main source of harmonic distortion. Since the value given by (4.19) is a peak-to-peak voltage, we add $V_c/2$ to one of the capacitor voltages and subtract the same value from the other capacitor voltage, and re-simulate the circuit. The change in the g_m of the output transistor is used as a measure of harmonic distortion. This value is tried to be kept below 5-10 percent, which results in THD values around 0.1-0.5 percent.

Table 4.4. Specifications of synthesized LNA

Spec.	**Target**	**Synthesis**	**HSPICE**
Gain (900MHz) (dB)	> 10	14.6	14.5
NF (dB)	< 2.5	1.95	1.9
s_{11} (dB)	< -10	-10	-10
Area (μm^2)	< 2000	1290	-
Current (mA)	< 10	7	7

The integrator's search space is relatively small, with twelve independent variables. Synthesis takes 2500($\pm$400) generations, 10($\pm$2) minutes. Table 4.5 compares the performances of the previously designed circuit and the synthesized circuit. Although the current is decreased from 16 μA to 6 μA, THD with a 10nApp 10 kHz input is not deteriorated noticeably; and yet there is a two-fold area and more than two-fold power gain.

Table 4.5. Specifications of synthesized HSCDCI

Spec.	**Target**	**Hand-design**	**Synthesis**
Gain (dB)	> 60	65	64.6
BW (kHz)	> 1	1.25	1.3
THD (%)	< 0.1	0.08	0.09
Area (μm^2)	< 500	264	124
Power (mW)	< 0.5	0.239	0.093

5. YIELD MAXIMIZATION

In analog circuit design, sizing the circuit at the global optimum may not always be a good choice in terms of maximizing yield. The design should be "centered" such that the circuit can tolerate process variations and mismatches. On the other hand, designing a very robust circuit results in poor power and area utilization. The trade-off between these criteria is performed through yield. This problem of maximizing the yield with minimum increase in area and power consumption has been an active research area.

Yield estimation mechanisms are intended for use either after optimization or during optimization. The well-known example of the post-optimization class of yield estimators is SPICE-based Monte-Carlo (MC) simulations. This method is accurate once enough simulations, usually on the order of tens of thousands, are performed. The high number of necessary simulations makes MC unsuitable to be directly included in the search loop, and MC should rather be used for verification purposes. A more recent example for post-optimization yield estimators is ROAD [32], ROAD extracts a quadratic polynomial and posynomial performance model using both transistor level simulation data and results of statistical analysis of process and environmental variations. The model is valid for the feasible region around the given initial solution and is used for fine-tuning.

In order to embed the yield estimator inside the search loop, a considerable speed-up over SPICE-based MC simulations is necessary. A method to accelerate yield estimation is response surface modeling. In this method, the dependence of performance metrics such as gain and bandwidth, to process variables is modeled. An advantage of response surface modeling is that direct modeling of performance metrics, rather than transistor responses, eliminates the need for time consuming simulation and post-processing steps. However, performance metrics present a more difficult surface

to be modeled. After modeling the response surfaces, yield is extracted by integrating the probability density function (PDF) over the feasible space. If linear models are used, as in [66] and [67], this integration is easier. However, linear models become inaccurate with the increasing process variations of today's submicron technologies. In order to overcome this problem, non-linear models are used at the cost of more difficult integration. In [68] and [69], quadratic models are used with various yield extraction mechanisms to get a speedup of 10-20 times compared to SPICE-based MC simulation.

An alternative and relatively simple method, which was experimented in the beginning of this study, is to perturb each process variable slightly during the synthesis run. By this way, SPASE calculates the response of a perturbed circuit. In the presence of this "process noise", the search algorithm converges to a more robust circuit, although it takes a longer time for convergence. Since this method is not yield-aware, synthesis results in over-conservative circuits with high values of area and power. For this reason, circuit perturbation method was abandoned and a yield-aware methodology based on piecewise cubic Hermite spline modeling of the response surface was developed in this study.

This chapter starts with a review of the process parameters where their variations are divided into local and global components. Methods to derive each component from the manufacturer given data are mentioned. Section 5.2 explains the reasoning behind the selection of cubic Hermite splines, elaborates the related mathematical background, and gives an example of modeling actual simulation data. Section 5.3 explains the integration of the Hermite model-based yield estimation mechanism in SACSES together with the implementation details of the Monte Carlo simulator. Section 5.4 demonstrates synthesis run results verifying the yield increase when the mechanism is used in the search loop.

5.1. Yield Related Process Parameters

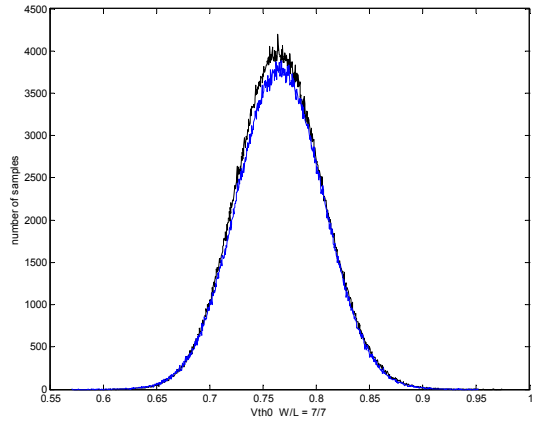
During the manufacturing process, transistor widths W, lengths L, gate oxide thicknesses t_{ox}, mobilities μ_0, substrate doping concentrations N_{SUB} and surface state densities N_{SS} can be assumed to be normally distributed. In the BSIM3 model for the target technology, we note that N_{SUB} and N_{SS} are not available. Inspecting the BSIM3 equations, it is found that channel doping, N_{ch}, can be used instead of N_{SS}. However, there is no direct substitute for N_{SUB}. Zero bias threshold voltage V_{th_0}, which is a related parameter, is available. Since V_{th_0} is dependent on the other process variables, it cannot be assumed to be normally distributed without further inspection. In order to determine the distribution of V_{th_0}, W, L, t_{ox}, μ_0 and N_{ch} are normally sampled, and V_{th_0} is evaluated using BSIM3v3 equations. Figure 5.1 shows the results of 10^6 samples for different W/L ratios. Also plotted on the figures is a Gaussian distribution having the same average and variance values as the V_{th_0} samples. These plots verify that for small process variations, we can assume V_{th_0} to be normally distributed.

Variations of these six process variables have local and global components:

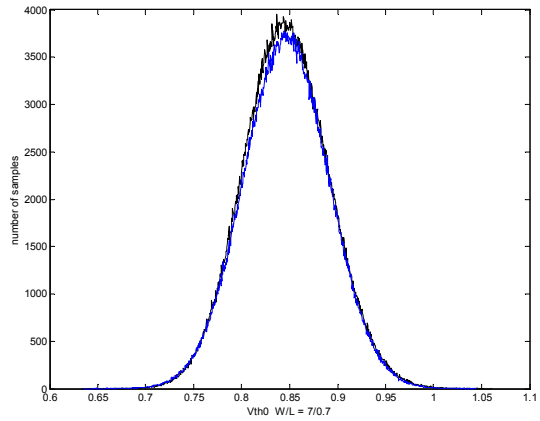
$$x_i = x_{i_0} + \Delta x_{local_i} + \Delta x_{global_i} \tag{5.1}$$

where x_{i0} is the nominal value of x_i, and Δx_{local_i} and Δx_{global_i} are its local and global variations, respectively. Global variations equally affect the entire wafer while local variations differ for each transistor. Depending on the amount of information given by the manufacturer, it may not be straightforward to deduce the local and global components of deviation. Local deviations can be derived from the mismatch parameters, which are given by the manufacturer in the form of a simplified Pelgrom model [70]:

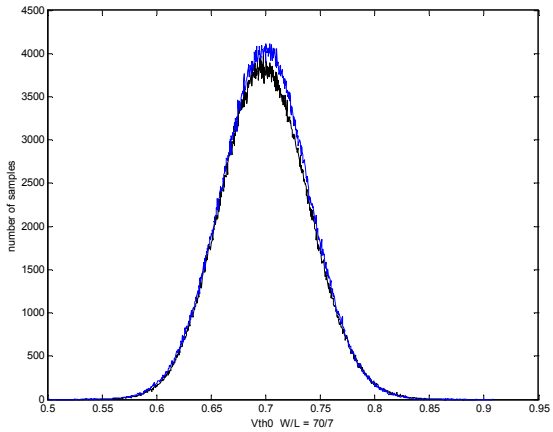
$$\sigma^2_{\Delta x_i} = \frac{A_p^2}{WL} \tag{5.2}$$



(a)



(b)



(c)

Figure 5.1. V_{th_0} distributions of 10^6 samples obtained from BSIM3v3 equations for W/L ratios of (a) 7μ m/7μ m (b) 7μ m/0.7μ m (c) 70μ m/7μ m

In order to determine global deviations, pass/fail parameters provided by the manufacturer are used. The parameters are assumed to be 3-sigma intervals and global deviations are calculated accordingly.

5.2. Response surface modeling

In order to develop a fast method to calculate yield, the effect of each process variable x_j on performance metric P_i is extracted separately. Since P_i is a performance metric, such as gain or bandwidth, no additional time consuming simulations are necessary once P_i is modeled. A direct method to calculate P_i from individual deviations is:

$$P_i = P_{i_0} + \sum_{j=1}^{N} \Delta P_{ij}\left(x_j\right) \tag{5.3}$$

where N is the total number of process variables, P_{i_0} is the nominal value of the i^{th} performance metric and $\Delta P_{ij}\left(x_j\right)$ is the deviation from its nominal value due to the variation of the j^{th} process parameter. It was observed that (5.3) gives over-pessimistic results due to the summing of each deviation. For this reason, an alternative method was used to calculate P_i where the nominal value, P_{i_0}, is updated using the root mean square (RMS) value of the individual deviations ΔP_{ij}:

$$P_i = P_{i_0} + \sqrt{\frac{\sum_{j=1}^{N} \Delta P_{ij}^2\left(x_j\right)}{N}} \tag{5.4}$$

In order to use (5.4) in the inner loop of a circuit synthesizer, we need to calculate ΔP_{ij} without transistor-level simulation, meaning that a model is necessary. The model to be used for this surface modeling should preserve the monotonicity and local extrema of the modeled surface. It should not introduce any other extrema. In this sense, high order polynomial models and rational functions can be problematic. The model should also be continuously differentiable and should have a high level of locality; i.e., a change in one sample should only affect the approximation near that

sample. Finally, the model should provide minimum error with minimum number of samples. In this study, several function types were experimented as candidates for surface modeling, including logarithmic and exponential functions, truncated Taylor series, polynomials and rational functions. Neither of the functions experimented were able to accurately model the response surface with a low number of samples. It was decided that the required accuracy can only be obtained by piecewise modeling of the response surface. Since linear and quadratic piecewise models have differentiability problems at the sample points, cubic splines have to be used. Among different cubic spline alternatives, piecewise cubic Hermite spline models satisfy all of the above requirements and are easy to generate and evaluate.

Given a set of $(N+1)$ sample points $(x_k, f(x_k)$ and $(N+1)$ derivatives $(x_k, f^{'}(x_k))$ of a function $f : [x_0, x_N] \rightarrow \mathrm{R}$ for $k = 0, \cdots, N$, the cubic Hermite spline interpolant on the interval (x_k, x_{k+1}) is:

$$H_k(t) = h_{00}(t)f_k + h_{10}(t)(x_{k+1} - x_k)f_k^{'} + h_{01}(t)f_{k+1} + h_{11}(t)(x_{k+1} - x_k)f_{k+1}^{'} \quad (5.5)$$

where the parameter t is obtained by normalizing the interval $(x_k, x_{k+1}))$ to $(0, 1)$:

$$t = \frac{x - x_k}{x_{k+1} - x_k} \quad (5.6)$$

and the Hermite basis functions h_{ij} are:

$$\begin{aligned} h_{00}(t) &= 2t^3 - 3t^2 + 1 \\ h_{10}(t) &= t^3 - 2t^2 + t \\ h_{01}(t) &= -2t^3 + 3t^2 \\ h_{11}(t) &= t^3 - t^2 \end{aligned} \quad (5.7)$$

If the derivatives $f^{'}(x_k)$ are not available or are expensive to compute, as in the case of circuit synthesis, they can be approximated using various techniques. The ap-

proximation method used in this thesis is given by (5.8) and preserves the monotonicity and local extrema of the sample points [71].

$$\tilde{f}'(x_k) = \frac{w_1 + w_2}{w_1/d_1 + w_2/d_2} \tag{5.8}$$

where,

$$\begin{aligned} w_1 &= 2h_1 + h_2 & w_2 &= h_1 + 2h_2 \\ d_1 &= \tfrac{f(x_k)-f(x_{k-1})}{h_1} & d_2 &= \tfrac{f(x_{k+1})-f(x_k)}{h_2} \\ h_1 &= x_k - x_{k-1} & h_2 &= x_{k+1} - x_k \end{aligned} \tag{5.9}$$

If the samples are monotonic, so is the approximation; and if a sample is a local extremum, so is the corresponding point in the approximation. Any local minima or maxima should not be introduced during approximation and Hermite splines satisfy this requirement. Another advantage of Hermite spline approximation is that it is extremely local. A change in one sample point only affects the splines in the neighboring two regions. This means that a steep non-linear region in the function can easily be approximated without increasing the error in the smoother regions of the function. Cubic Hermite spline interpolation is exact at the sample points; the first order derivatives at the sample points are also exact. The error introduced is $O(h^4)$ [71], meaning that the error can be greatly reduced by increasing the number of samples.

Figure 5.2 compares simulation data with interpolated data for the change of the gain of the BTS OPAMP circuit of Figure 4.5, with respect to the width of one of the input transistors. The number of samples is 11. Although the transistor width spans a very small range, gain is very sensitive to this parameter and takes a wide range of values. While the data interpolated with cubic Hermite splines show good agreement with the simulated data, results of cubic spline interpolation, which is a widely used type of piecewise interpolation, suffer from unacceptably high errors in

some regions. In addition, cubic spline interpolation fails to preserve the monotonicity and local extremity information gathered from the samples, which is not suitable for use in a search algorithm.

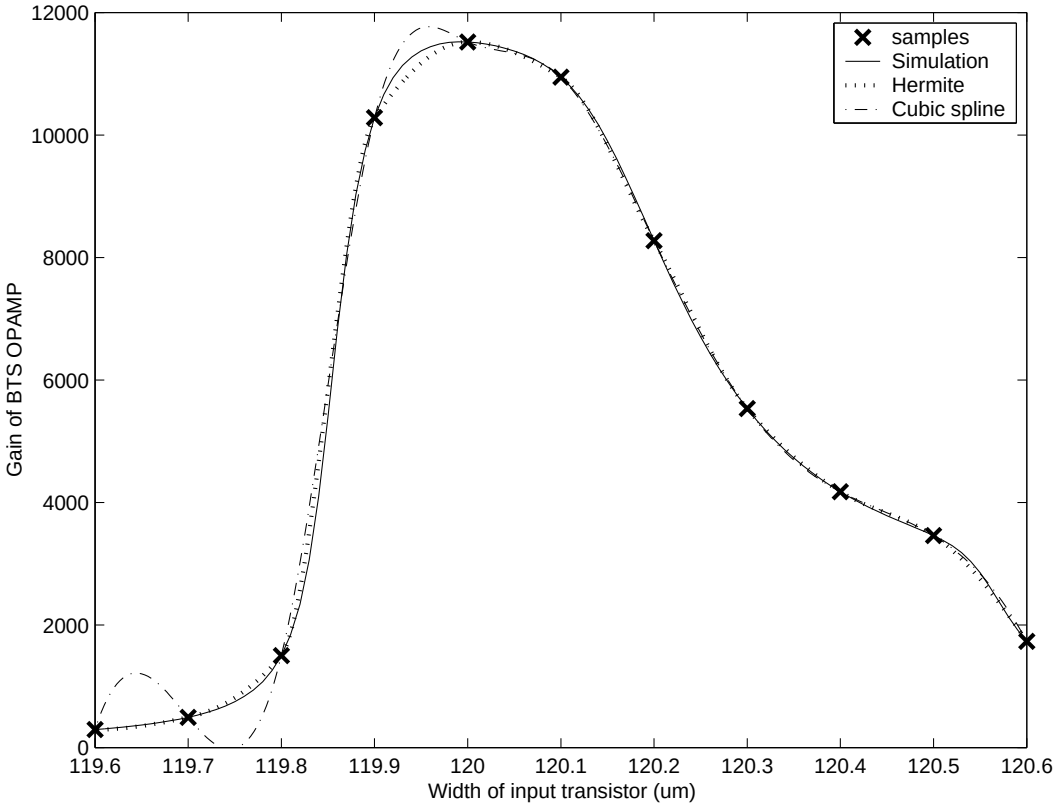


Figure 5.2. Comparison of simulation data with interpolation data

5.3. Yield Estimation Methodology

In order to estimate yield, a MC simulator is implemented and added to the SACSES synthesis environment. The MC simulator perturbs each process parameter according to (5.1), evaluates the resulting circuit and checks the user given constraints for each performance metric to evaluate yield. In the evaluation stage, MC simulator can use either SPASE or Hermite models. During the development of sampling scheme for Hermite models, it was noted that common mode variations in the parameters of transistor pairs have minimal effect on circuit performance while difference mode variations cause rapid deterioration of performance metrics. Correspondingly, estimating P_i using (5.4) gives over-pessimistic results. In order to overcome this problem, common and difference mode perturbations in the transistor pairs are separately approximated,

requiring an extra set of samples for each transistor pair. Hence, the complete piecewise cubic Hermite model requires a total of:

$$n_{total} = (n_{tr} + n_{pair}) n_{var} n_s \tag{5.10}$$

samples where n_{tr} is the number of transistors, n_{pair} is the number of transistor pairs, n_{var} is the number of process variables and n_s is the number of samples evaluated for each $\Delta P_{ij}(x_j)$.

In order to compare the accuracy of SPASE and Hermite model-based yield estimation methodologies, two BTS OPAMP circuits were tested with 10^4 MC runs. The first circuit is the result of a previous SACSES run without any design centering effort. The second circuit is centered with minimal effort by the designer by increasing some of the transistor widths and lengths. By using two different sets of acceptance thresholds, a low and a high yield results was obtained from each BTS OPAMP circuit. The resulting four MC yield estimation results and computation times on a 2GHz PC using SPASE and Hermite models are given in Table 5.1.

For the Hermite models, $\Delta P_{ij}(x_j)$ of (5.4) is sampled at $x_j = x_{j0} \pm 3\sigma$, meaning that $n_s = 7$. For the BTS OPAMP circuit, $n_{tr} = 12$, $n_{pair} = 2$, and $n_{var} = 6$. Out of the seven samples required, one is readily available from nominal design. So, from (5.10) $n_{total} = (12+2) \cdot 6 \cdot (7-1) = 504$ extra simulations are required for model development. This process takes 13 sec and is included in the computation times given in Table 5.1. It should be noted that, during sampling, 25.8 msec is required per simulation, while SPASE based MC analysis requires 49 msec per simulation. The reason for this time per simulation difference is that the acceleration mechanisms of SPASE favor circuits with parameters close to each other. During modeling, only one parameter is perturbed at a time, and SPASE requires almost half the iterations for DC convergence and AC response tracing in BW and PM calculations. Another important point is that the response surface modeling time of 13 sec dominates the overall yield estimation time of

15 sec. In other words, even with the use of the MC method with 10^4 runs to integrate the PDF and extract yield, model generation time takes more than 85% of the total yield estimation process.

Table 5.1. Comparison of yield estimation mechanisms

Circuit	**HSPICE**	**SPASE**	**Hermite-models**
BTS-1-Low	61.9%	61.7%	59.8%
		490 sec	15 sec
BTS-1-High	71.1%	71.2%	70.2%
		482 sec	15 sec
BTS-2-Low	82.7%	82.8%	79.1%
		473 sec	15 sec
BTS-2-High	89.9%	89.8%	86.5%
		471 sec	15 sec

Inspecting Table 5.1, we note that if SPASE is used for evaluation, results comparable in accuracy with those of commercial MC simulators are obtained. However, Hermite models give a few percent lower yield estimation results. The main reason for this inaccuracy is the separation of the effects of each process variable, rather than Hermite model interpolation inaccuracy. Despite this inaccuracy, the model successfully differentiates between a relatively low yield and high yield circuit and is more than 30 times faster. These two features make Hermite models still suitable for inclusion in the search loop.

When yield estimation is embedded in the search loop, it is not necessary to calculate yield at the early stages of evolution. In this premature stage of evolution, the circuits do not satisfy even the biasing constraints. Instead, yield estimation should be introduced only when the cost drops below a certain value. On the other hand, too late an introduction, after all of the constraints are satisfied,causes minimal yield improvement because the synthesizer misses other constraint-satisfying regions. Our

experiments show that the optimal stage for starting yield estimation is when the synthesizer is close to finding a circuit that satisfies the given constraints. This stage, which roughly corresponds to the last third of evolution, is detected by a control routine that continuously inspects the best individual's status, and once yield estimation is triggered, a yield-dependent term is added to the nominal cost of each individual, C_{i_0}:

$$C_i = C_{i_0} + w_{yield}(Y_i - Y_t) \tag{5.11}$$

where w_{yield} is the user-given weight for yield, and Y_i is the yield of the i^{th} individual, and Y_t is the target yield value. During the final stage of evolution, when performance targets are reached and power and area optimization is performed, higher accuracy for yield is needed and the pessimistic results of the Hermite model-based MC simulations become inadequate. In order to overcome this problem, the best individual of each generation is also evaluated with SPASE-based MC simulation of 10^3 runs, and all other yield estimations are scaled accordingly.

5.4. Synthesis Results

In order to evaluate the usefulness of the proposed yield estimation mechanism, two synthesis runs are performed, with and without the Hermite model based yield estimation mechanism. Table 5.2 summarizes the results. All of the given specifications are included in the cost function and optimized throughout the synthesis process. The values in the target column are also taken as the acceptance limits for yield calculation. In order to get accurate yield values, final yield is calculated using 10^6 SPICE-based MC runs. When the two circuits are compared, it can be noted that the performance metric values near the acceptance limits, such as gain, are improved while those that have a high margin, such as bandwidth, are relaxed. Correspondingly, yield is increased from 64.8% to 91.2%. Area and power values are also increased. The synthesis times are 18 minutes without yield estimation, and 27 hours with yield estimation. The long yield-aware synthesis time can be improved by taking advantage of the algorithms used. First, the ES algorithm is inherently parallelizable by evaluating each individual on a different node. Second, the 504 extra simulations needed for model development

can also be distributed. Since the number of these extra simulations is fixed and the corresponding circuits are completely independent from each other, synthesis time can be decreased almost linearly by parallelization.

Table 5.2. Specifications of the synthesized BTS OPAMP with and without yield estimation

Specification	**Target**	**w/o Yield**	**w/ Yield**
A_0 (dB)	> 75	75.5	76.5
BW (kHz)	> 10	14.4	13.1
r_{out} (kΩ)	< 50	28.5	31.3
V_{os} (mV)	< 10	2.8	1.3
PM	$> 60°$	69°	68°
CMRR (dB)	> 80	94	95
Power (mW)	< 5	< 1.92	2.35
Area (μm^2)	< 30000	21600	27300
Yield (%)	90	64.8	91.2

6. LINK BETWEEN SYSTEM and CIRCUIT LEVEL AUTOMATION

During system level synthesis, the DA tool needs power and area values of the blocks it is using in the design. Without this knowledge, the tool can converge to an unrealizable or a sub-optimum solution. Since it is very time consuming to run circuit synthesis in the inner loop of the system level synthesizer, the needed power and area values are usually obtained from approximate models included in an auxiliary DA tool, performance estimator (PE). Performance estimation necessitates calculating power and area given other performance criteria such as gain and bandwidth. In order to be fast enough, this calculation should not involve actual synthesis of the building block, meaning that power and area should be calculated without circuit sizing. This makes developing an accurate yet fast PE a very difficult task. PEs are also usually topology dependent and require design expertise to develop. In this thesis, instead of using PEs, optimization algorithms of system and circuit level DA tools are suggested to run hierarchically. Genetic algorithms and evolutionary strategies are known for their fast convergence to the near-optimal region and slow fine-tuning to reach the optimal point. So, GA or ES can be used for both levels and synthesis level can be switched after coarse optimizations. This approach takes advantage of the the fast initial convergence of ES algorithms while avoiding the time consuming fine-tuning stage.

The proposed design flow with hierarchical genetic algorithm (HGA) is shown in Figure 6.1. Optimization is started at system level design tool and truncated after a certain number of generations. Then, the specifications of the circuit-level building blocks of each individual are transfered to the circuit level design tool, and circuit level synthesis is started. Circuit synthesis is truncated after the cost value decreases below a certain level. At this stage, actual cost values of individuals, including the power and area metrics as well as system performance, are approximately obtained. Then, upper module is re-invoked with the approximate costs of individuals. During the final

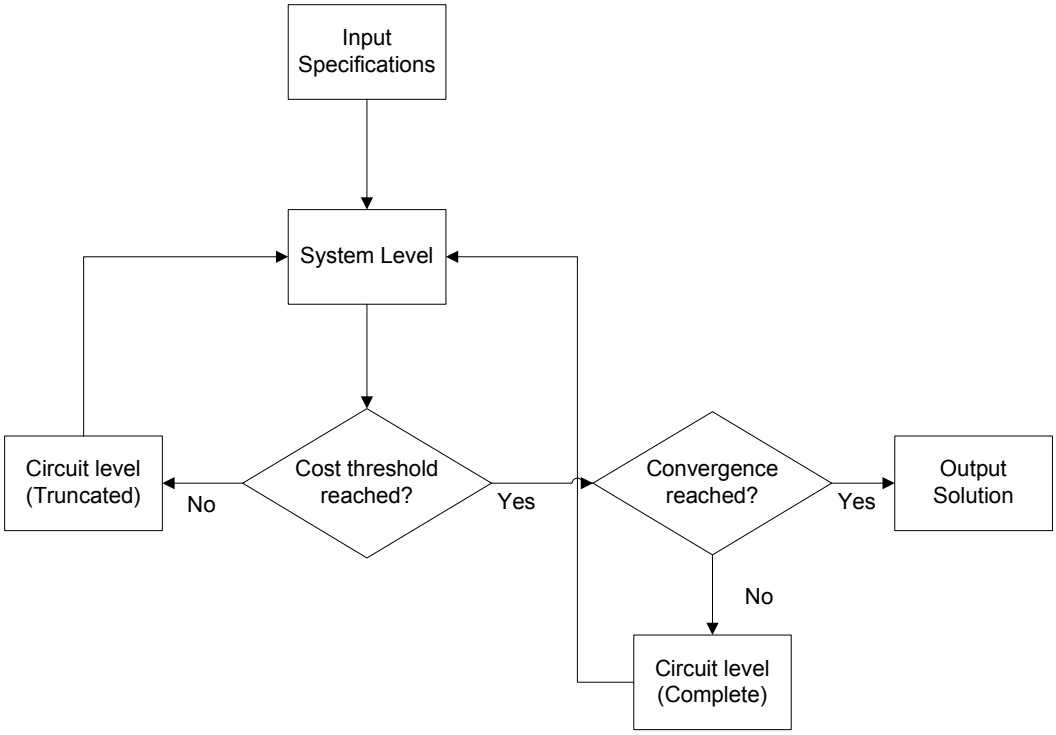


Figure 6.1. Proposed hierarchical design flow

stage of optimization, when the system level cost reaches the lower threshold, complete circuit level synthesis are run in order to get accurate cost values of individuals.

6.1. Synthesis Example

In this Section, synthesis results of a 3^{rd} order Butterworth filter with two different implementations are given. The two implementations mainly differ in their system level performance evaluation mechanisms. The first implementation uses derived equations based on BTS OPAMP macro-model parameters while the second implementation uses transistor-level simulation for system level performance evaluation. Both implementations use transistor-level simulation for circuit level performance evaluation.

Butterworth filter design is relatively simple and a widely studied subject. However, the main objective of this section is to test the applicability of the proposed hierarchical genetic algorithm (HGA) and compare its performance with that of non-hierarchical approaches. Hence, the hardness of the problem is not a primary concern.

6.1.1. HGA based on macro-models and equations

In co-operation with [8], a 3^{rd} order Butterworth filter is synthesized using three different approaches. The first approach is based on an equation-based performance estimator, the second one is the proposed hierarchical algorithm, and the last one invokes the circuit level design tool at every system level iteration. Overall synthesis times and performances of the resulting circuits are compared.

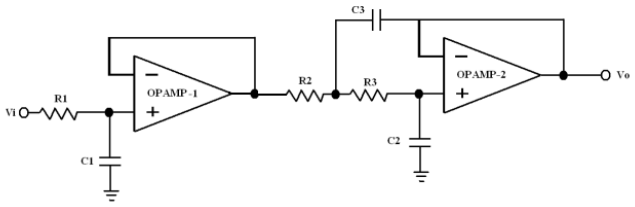


Figure 6.2. 3^{rd} order low pass Butterworth filter with Sallen-Key topology

The 3^{rd} order Butterworth filter, shown in Figure 6.2, utilizes the Sallen-Key topology and has the following transfer function when ideal OPAMPs are used:

$$H(s) = \frac{V_0(s)}{V_i(s)} = \frac{\omega_c^3}{s^3 + 2\omega_c s^2 + 2\omega_c^2 s + \omega_c^3} \tag{6.1}$$

where,

$$\omega_c = \frac{2}{C_1R_1 + C_2R_2 + C_3R_3} \tag{6.2}$$

Finite gains and output impedances of the OPAMPs are accounted for using the OPAMP model of Figure 6.3. Finite bandwidths of the OPAMPs are accounted for by requiring that the OPAMP bandwidth should be k times the required filter bandwidth. The empirical value of $k = 1.5$ is used in this synthesis example. Using this non-ideal

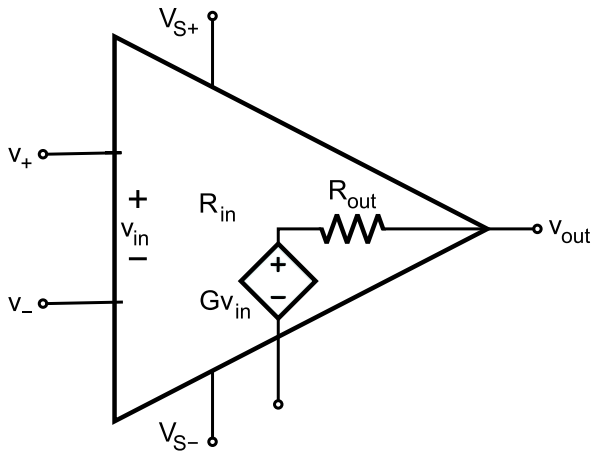


Figure 6.3. Non-ideal OPAMP model used in the synthesis example

model, the transfer function takes the form:

$$H(s) = \frac{V_0(s)}{V_i(s)} = \frac{K_3 Z_2 s^2 + K_3 Z_1 s + K_3 Z_0}{s^3 + \frac{K_2}{K_3} s^2 + \frac{K_1}{K_3} s + \frac{K_0}{K_3}} \tag{6.3}$$

where the coefficients K_i depend on the external resistor and capacitor values as well as finite gains and output resistances of OPAMPs. Symbolic expressions for K_i are calculated using the symbolic analyzer SAPWIN [72]. Comparing the transfer function equations of (6.1) and (6.3), we note that parasitic zeroes are added and the cut-off frequency, f_c, can be obtained in three ways:

$$\begin{aligned} f_{c1} &= \frac{K_2}{4\pi K_3} \\ f_{c2} &= \frac{1}{2\pi}\sqrt{\frac{K_1}{2K_3}} \\ f_{c3} &= \frac{1}{2\pi}\sqrt[3]{\frac{K_0}{K_3}} \end{aligned} \tag{6.4}$$

Ideally, the three cut-off frequencies are equal. Otherwise, circuit response deviates from the ideal Butterworth characteristic. In order to form a cost function using (6.4), first, each center frequency is normalized using:

$$k_{fi} = \frac{f_{ci} - f_{c,target}}{f_{c,max} - f_{c,min}} \tag{6.5}$$

where $i = 1, 2, 3$, $f_{c,max}$ and $f_{c,min}$ are the minimum and maximum acceptable values for the center frequency calculated from the user given target frequency, $f_{c,target}$ and tolerance value. Then, the cost of other performance metrics, P_j, consisting of power, area, and ratios of maximum and minimum external resistors and capacitors are calculated from:

$$k_{P_j} = \frac{|P_j - P_{j,good}|}{|P_{j,good} - P_{j,bad}|} \tag{6.6}$$

which is similar to the normalization function of (4.3). Finally, the overall cost, C, is calculated as a weighted sum of the individual cost terms:

$$C_{system} = w_{fc}(k_{f1}^2 + k_{f2}^2 + k_{f3}^2) + w_p k_p + w_a k_a + w_R k_R + w_C k_C \tag{6.7}$$

where w_{fc}, w_p, w_a, w_R, and w_C are the weights for the cut-off frequency, power, area, resistor and capacitor ratios, respectively.

This cost function was embedded in a standard genetic algorithm to form the system level synthesizer [8]. Three different possibilities for integration with the circuit level were tested. In all of the three cases, the population size for the system level was 20 and target $f_c = 10$ kHz with 1% tolerance. In the first approach, the area and power values of individuals were estimated by simplified equation-based models created from previous synthesis data of 10^6 BTS OPAMPS. In the second approach, system level optimization was truncated every 20 generations and block specifications of the 20 individuals were transfered to SACSES. SACSES orders the block specifications according to the FOM of GBW/r_{out}, and starts from the block having lowest FOM. Once the cost for this block drops below 3, SACSES switches to the next block in the ordered list. By this way, synthesis is not started from the beginning each time, and the overall synthesis time is reduced. After all of the blocks are synthesized, their area and power values are returned to the system level synthesizer and the iteration restarts. If the system level cost drops below 2, SACSES is instructed to perform complete synthesis of each building block. The third approach is direct embedding of SACSES

in the system level synthesis loop, meaning that 20 SACSES runs are performed for each system level iteration. Ordering the blocks according to increasing FOM values is again performed. The cost thresholds of 2 for the system level, and 3 for the circuit level was determined by performing independent system and circuit level synthesis runs before starting the integrated algorithm.

Table 6.1 summarizes the results of the three approaches. It is clear that the inaccurate equations lead to under-estimation of power and area in the first approach. Although this is the fastest approach, it necessitates an OPAMP with very high area and power consumption. On the other hand, the HGA and direct embedding methods provide feasible OPAMP blocks. The characteristics of the synthesized filters are also similar. However, the total synthesis time for the HGA approach is 60 times lower than the direct embedding method. This means that considerable speed-up is possible with the proposed HGA approach without sacrificing circuit performance.

Table 6.1. Synthesis results for integration with three different approaches

Specification	**Equation-based**	**HGA**	**Direct embedding**
Filter f_c (kHz)	9.503	9.930	9.975
Filter R_{max}/R_{min} (Ω/Ω)	251/154	240/98	254/42
Filter C_{max}/C_{min} (nF/nF)	212/54	240/45	511/62
OPAMP A_0 (dB)	74.5	68	70.5
OPAMP BW (kHz)	14.3	14.9	15.0
OPAMP r_{out} (kΩ)	3.1	1.2	4.5
OPAMP Power (mW)	45	2	2
OPAMP Power Estimation (mW)	2	2	2
OPAMP Area (μm^2)	53000	34000	34000
OPAMP Area Estimation (μm^2)	36000	34000	34000
Number of generations	47	61	71
Synthesis time (sec)	923	2143	128456

6.1.2. HGA based on transistor-level simulation

Although the HGA method presented in the previous sub-section high synthesis speed, the method relies on macro-models at the system level. Macro-models require considerable amount of initial effort and are topology dependent. Also, modeling higher order effects is difficult, if not impossible, with macro-models. In order to overcome this problem, a generalized version of the proposed HGA method, where the whole circuit is simulated at the transistor level, is implemented. A (30+20)-ES scheme is used in both system and circuit levels. The synthesis flow can be summarized with the following steps:

- Synthesis starts at the circuit level, where BTS OPAMPs with moderate specifications are synthesized. For instance, the target minimum values for gain and bandwidth are $A_0 = 2000$ and $BW = 5000$ Hz. This initial synthesis is truncated when the average cost of the population drops below 5.
- The entire population of 50 individuals is fed to the system level synthesizer. From each circuit level individual consisting of a sized BTS OPAMP, a system level individual representing the 3^{rd} order Butterworth filter of Figure 6.2 is formed by using two identical OPAMPS and adding external resistor and capacitors.
- Instead of performance equations like (6.4), filter performance of each system level individual is evaluated by using transistor level simulation. For this purpose, frequency and phase responses of the filter is sampled at 100 different points from 1 Hz to 1 MHz and the sum of squared differences between these samples and samples from an ideal 3^{rd} order Butterworth filter having the same gain is evaluated. In addition to the summed squared difference, overall power, area values and maximum resistor and capacitor ratios are included in the cost function.
- In the system level, only the external resistor and capacitor values are altered and synthesis is truncated after 20 generations. Surviving BTS OPAMPS, together with their resulting system cost value, are passed to the circuit level synthesizer.
- If the circuit level synthesizer continues evolution with the previous target performance values, it is clear that information gained from the system level, i.e. suitability of each BTS OPAMP for the Butterworth filter, will be lost within a

few generations under the influence of the selection operator. In order to overcome this problem, circuit level synthesizer calculates a weighted average, P_i^{av}, for each OPAMP specification according to:

$$P_i^{av} = \frac{\sum_{j=1}^{N} P_i^j / C^j}{\sum_{j=1}^{N} 1/C^j} \tag{6.8}$$

where N is the number of individuals, P_i^j is the value for the i^{th} performance metric of the j^{th} individual, and C^j is the system level cost value of the j^{th} individual. Then, the target value for the i^{th} performance metric, T_i is set 50% higher than P_i^{av}, unless the resulting T_i value is beyond the user-given technology limit. By this way, performance targets are adapted according to the system level performance of the individuals.

- Circuit level synthesis, with new target performance criteria, is truncated after 200 generations or if the average cost of the population drops below 5, the resulting OPAMPs are passed to the system level, and the loop restarts. Similar to the macro-model based synthesis, in the later stages of evolution, the system level synthesizer informs the circuit level synthesizer that the system level performance criteria are about to be met. In this case, circuit level synthesis is run completely.

Table 6.2 gives the synthesis results of the above implementation for a target of f_c = 10 kHz with 1% tolerance. Also given in the table are the results of non-hierarchical transistor level synthesis, in which all of the component values of the Butterworth filter are optimized together. Similar to the previous subsection, a considerable speedup, 15 times for this synthesis example, is obtained by using hierarchical synthesis techniques, without sacrificing circuit performance. Compared to the macro-model based approach, transistor-level simulation at the system level requires about 7.5 times higher synthesis time. However, the time required for the generation of performance equations and macro-models is not included in the macro-model based synthesis time. As a result, unless the macro-models and equations are readily available or the simulator is not

capable of simulating the system level circuit, the transistor-level approach should be preferred.

Table 6.2. Comparison of synthesis results of hierarchical and flat synthesis approaches

Specification	**HGA**	**Flat**
Filter f_c (kHz)	10.015	10.012
Filter R_{max}/R_{min} (Ω/Ω)	238/96	223/85
Filter C_{max}/C_{min} (nF/nF)	232/41	243/45
OPAMP A_0 (dB)	70.4	70.2
OPAMP BW (kHz)	15.1	15.0
OPAMP r_{out} (kΩ)	2.8	3.2
OPAMP Power (mW)	2.2	2.1
OPAMP Area (μm^2)	33900	33750
Number of generations	1100system + 11724circuit	23600
Synthesis time (sec)	(11880+4212)=16092 (4.5h)	253550 (67h)

7. LINK BETWEEN CIRCUIT and LAYOUT LEVEL AUTOMATION

After circuit sizing, the layout level DA tool, TOLAS (Tool Oriented Layout Automation System), which is the subject of another thesis study, generates the chip layout using the circuit netlist taken from SACSES. TOLAS [73] is a framework coded in JAVA and it consists of tools and databases. Databases are used to handle the data such as netlist and layout. Databases are used as interfaces between tools, where tools are code routines for automation purposes such as device generation or GDSII (Graphic Database System) exporting. Due to the fact that the framework is based on instances of tools, the system is named tool oriented.

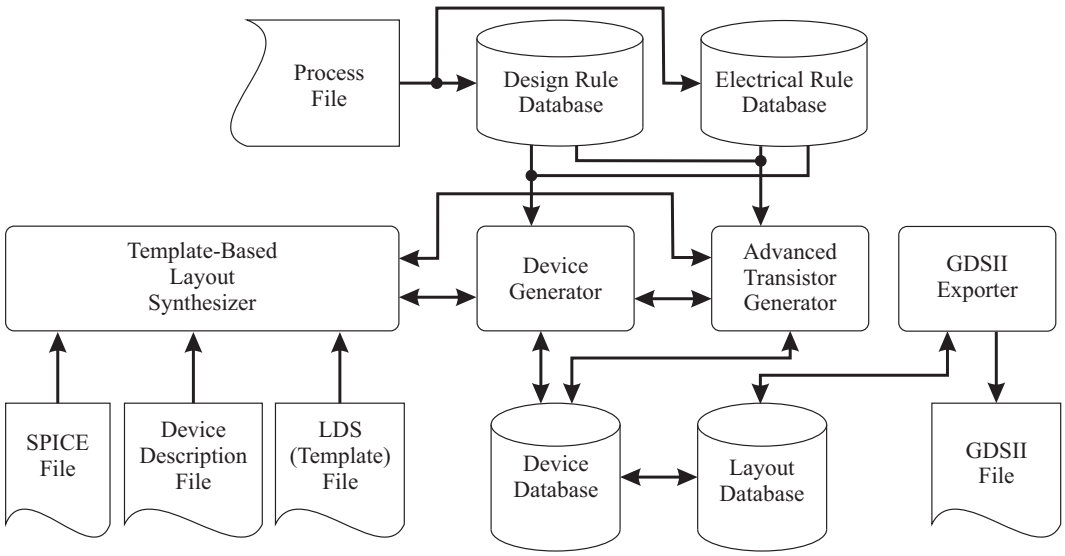


Figure 7.1. Framework of the layout generator, TOLAS

A template based layout generator is implemented in this framework as depicted in Figure 7.1. In this implementation, layout database handles the layout in a hierarchical structure, device database records the devices and their ports, and process rules are stored in design and electrical rule databases. The device generator constructs devices for the AMS 0.35 μm technology; whereas the Advanced Transistor Generator

constructs transistors and performs the routing of fingered and interdigitized transistors. Finally, the GDSII Exporter writes the layout database into a file. The main tool, Template Based Layout Synthesizer, constructs a layout based on an input template using linear programming. The template file is written in Layout Description Script (LDS), which defines constraints between devices as well as routing. Alternatively, this template may be constructed through a graphical user interface (GUI). While current version of TOLAS requires a user-given template, automated template generation is under development.

During the layout generation process, inevitable layout parasitics cause some deviation from the target performance values. Although TOLAS tries to minimize these parasitics, attaining the target performance values is not guaranteed. In other words, layout parasitics, although minimized, can still cause unacceptable deviations from target values. In such a case, a feedback loop to the circuit level synthesizer is proposed as shown in Figure 7.2.

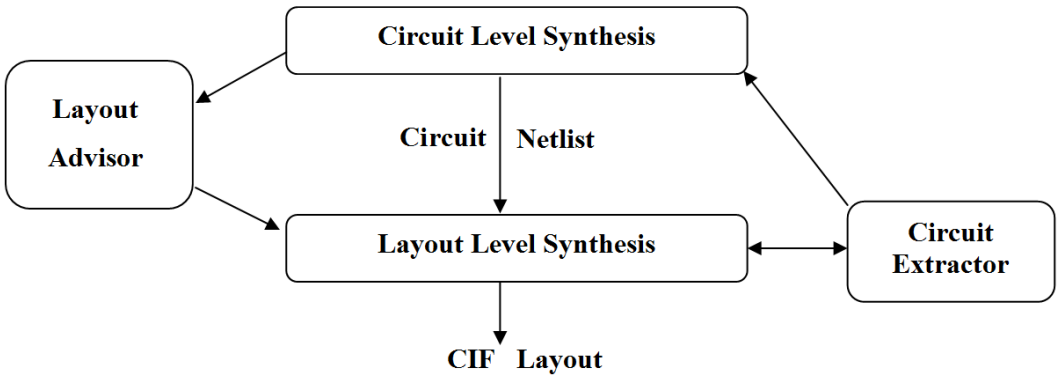


Figure 7.2. Design automation flow with proposed feedback of layout parasitics into circuit level

Using the GDSII file generated by TOLAS, a circuit netlist is extracted that includes layout parasitics as well as partitioned large transistors. Since the implementation of an extractor is beyond the scope of this thesis, the extractor of the Mentor Graphics Calibre is used. The proposed feedback enables the circuit synthesizer to

take layout parasitics into account. Since parasitics are minimized by TOLAS, the optimal solution should be near to the initial one. The extracted parasitics are not altered during circuit level synthesis and in order to stay within the validity region of the extracted parasitics, component values should not be changed too much during resizing at the circuit level. Hence, we should now search for the local optimum around the starting point. Consequently, the ES algorithm is now restarted with a low temperature value, so that only local search is performed. Mutation step-size limit and ranges of variables are also decreased. The resulting smaller search space shortens the synthesis time, which is already lengthened due to layout parasitics and partitioned transistors. Depending on the amount of layout parasitics, re-iterations maybe necessary for convergence.

7.1. Synthesis Example

In order to test the proposed feedback loop, a BTS OPAMP was synthesized and run through the loop. Table 7.1 shows the target and attained performance values after the initial SACSES run. A high gain was targeted to exaggerate the effects of parasitic capacitors through the Miller effect. The first synthesis run, without any layout parasitics, is completed after 2900 generations, 17 minutes.

Table 7.1. Specifications of the BTS OPAMP after the first run of SACSES

Specification	**Target**	**Synthesis**	**HSPICE**
A_0 (dB)	> 85	88	88
BW (kHz)	> 5	5.7	5.7
r_{out} (kΩ)	< 10	9.2	9.2
V_{os} (mV)	< 25	18.2	19.8
PM	$> 50°$	52°	51°
CMRR (dB)	> 80	82	82
SR (V/μs)	> 5	7.3	7.4
Power (mW)	< 20	13.7	13.7
Area (μm^2)	< 100000	55480	-

The synthesized BTS OPAMP was fed to TOLAS together with the layout template given in Figure 7.3. The resulting layout, generated in 4 seconds, is shown in Figure 7.4. This layout was extracted using the Mentor Calibre package. Table 7.2 compares the HSPICE simulation results of the pre-layout and post-layout BTS OPAMP circuits for the first iteration. Inspecting the table, we note that parasitic capacitances shift the phase margin below the acceptance threshold. Parasitic layout capacitances are mainly due to the routing between the transistors and are on the order of femto Farads. Hence, they have limited effect on circuit bandwidth. On the other hand, layout parasitics can change the unity gain frequency, which is around 60MHz in this circuit, and alter the phase response at high frequencies. As a result, phase margin is shifted. From Table 7.2, we also note that the output offset voltage, V_{os}, is also shifted to the unacceptable region. This is due to transistor partitioning, which slightly alters transistor biasing and causes changes in the offset voltage.

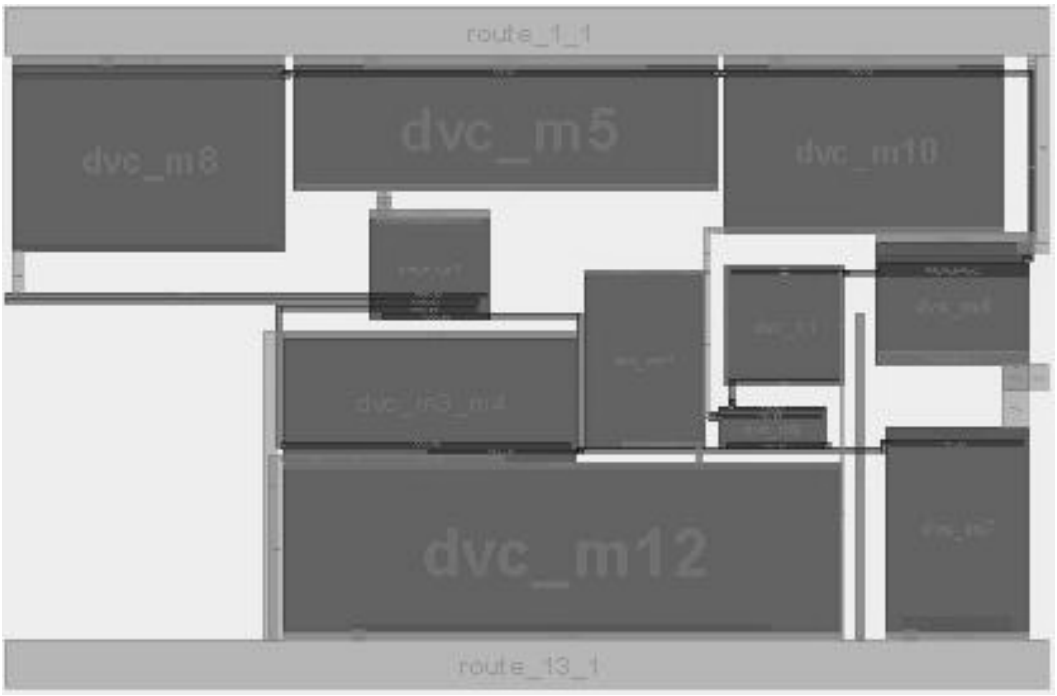


Figure 7.3. Layout template for the BTS OPAMP

In the second iteration, the netlist obtained from Calibre, which includes layout parasitics and transistor partitioning information, is fed to SACSES. This time, SACSES searches within 10% of the initial circuit parameters. Interconnect and cross-coupling parasitics remain unchanged since their updated values cannot be calculated

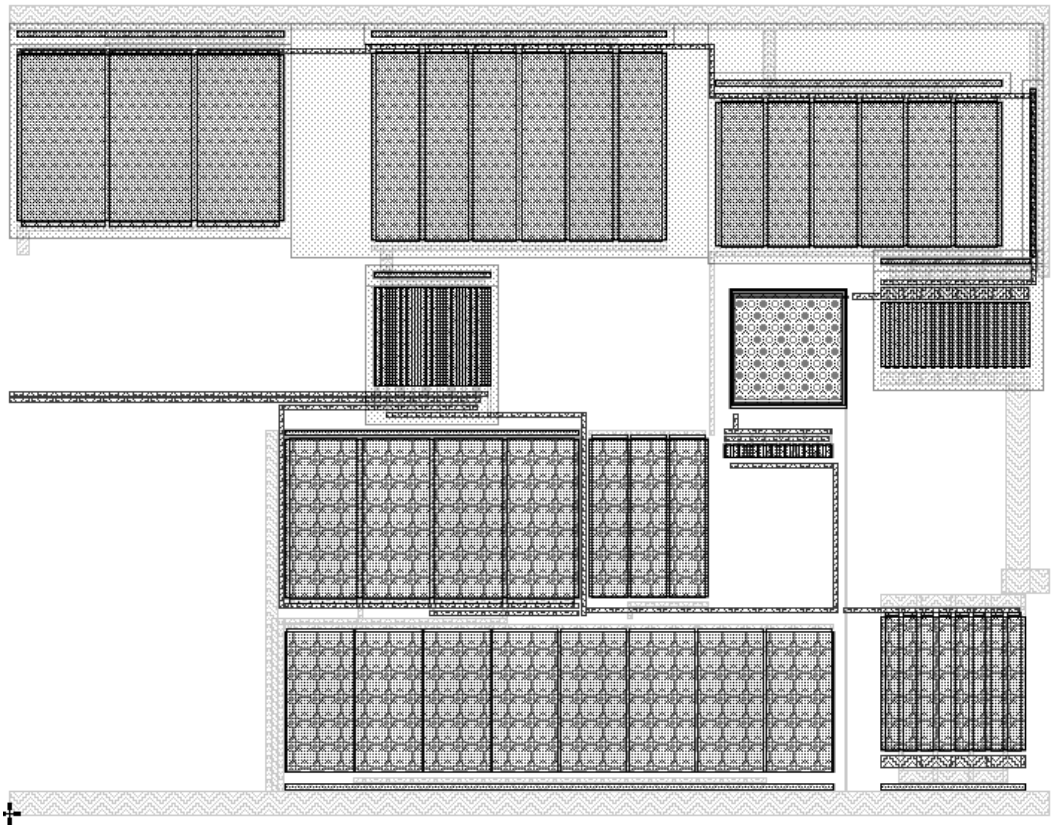


Figure 7.4. Layout of the BTS OPAMP circuit after the first iteration

Table 7.2. Specifications of the pre-layout and post-layout BTS OPAMP circuits after the first iteration

Specification	**Target**	**Pre-layout**	**Post-layout**
A_0 (dB)	> 85	88	88.5
BW (kHz)	> 5	5.7	5.2
r_{out} (kΩ)	< 10	9.2	9.7
V_{os} (mV)	< 25	19.8	138
PM	$> 50°$	51°	45°
CMRR (dB)	> 80	82	83
SR (V/μs)	> 5	7.4	7.3
Power (mW)	< 20	13.7	13.7
Area (μm^2)	< 100000	55480	64484

without running TOLAS inside the search loop. Moreover, since transistor sizes are not altered too much, the layout and related parasitics are not expected to be very different. On the other hand, the drain and source capacitances are scaled according to the widths of the transistors using the extracted AD, AS, PD and PS values. Source and drain resistances are also scaled using NRD and NRS values. This SACSES run requires 500 generations and takes 190 minutes. Although the narrow search space requires fewer generations, increase in the number of nodes from 41 to 193 slows down the simulations and lengthens the synthesis time. Table 7.3 compares the HSPICE simulation results of the pre-SACSES and post-SACSES BTS OPAMP circuits in the second iteration. From the table, we note that output offset and phase margin values are again within target specifications. This time, since the layout parasitics were taken into account, the performance specifications are not expected to deteriorate too much after layout generation.

Table 7.3. Specifications of the pre-SACSES and post-SACSES BTS OPAMP circuits in the second iteration

Specification	**Target**	**Pre-SACSES**	**Post-SACSES**
A_0 (dB)	> 85	88.5	88.1
BW (kHz)	> 5	5.2	5.2
r_{out} (kΩ)	< 10	9.7	9.3
V_{os} (mV)	< 25	138	3.9
PM	$> 50°$	45°	52°
CMRR (dB)	> 80	83	83
SR (V/μs)	> 5	7.3	7.3
Power (mW)	< 20	13.7	13.6
Area (μm^2)	< 100000	64484	64420

Figure 7.5 shows the layout after the second iteration and Table 7.4 compares the HSPICE simulation results of the pre-layout and post-layout BTS OPAMP circuits after the second iteration. The deviation from the pre-layout specifications is minimal and the final circuit is within acceptable performance criteria. This means that two

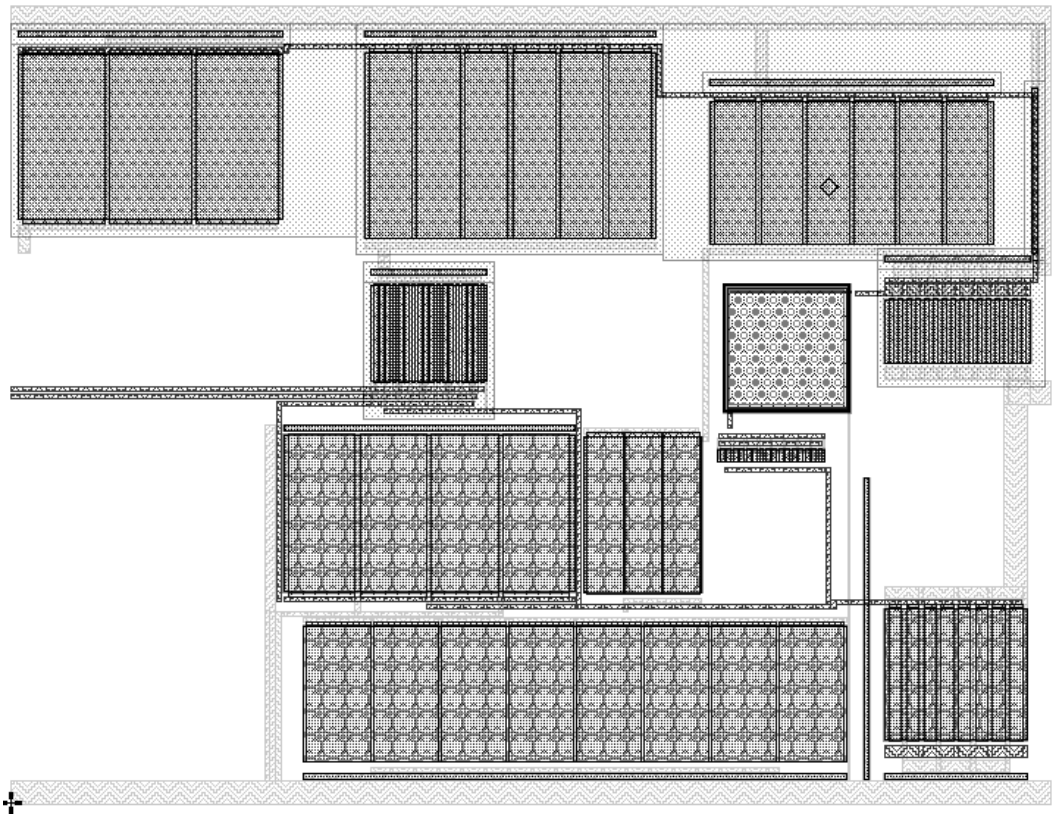


Figure 7.5. Layout of the BTS OPAMP circuit after the second iteration

Table 7.4. Specifications of the pre-layout and post-layout BTS OPAMP circuits after the second iteration

Specification	**Target**	**Pre-layout**	**Post-layout**
A_0 (dB)	> 85	88.1	88.1
BW (kHz)	> 5	5.22	5.19
r_{out} (kΩ)	< 10	9.3	9.3
V_{os} (mV)	< 25	3.9	4.0
PM	$> 50°$	52.1°	51.7°
CMRR (dB)	> 80	83	83
SR (V/μs)	> 5	7.3	7.3
Power (mW)	< 20	13.6	13.6
Area (μm^2)	< 100000	64420	64430

iterations through the feedback loop is enough for this BTS OPAMP circuit. It should be mentioned that the area values given in the tables are total transistor areas; with the addition of guard rings, routing areas and unused spaces, the total silicon area for the final run is 152145 μm^2. The overall synthesis time, including layout extraction in Calibre, is 212 minutes.

8. CONCLUSION and FUTURE WORK

A simulation-based analog circuit synthesis methodology and its implementation, SACSES, was presented. With the use of simulation-based approach, SACSES is topology independent and requires minimal initial effort. SACSES uses an in-house circuit simulator, SPASE, which is developed according to the needs of circuit synthesis. Accuracy of SPASE was tested with BTS OPAMP simulations and was found to be comparable to that of HSPICE. Advantages of the acceleration mechanisms of SPASE were also validated. It was shown that taking the previous DC solution as the starting point of the next DC analysis more than halves the number of iterations required for convergence. Similar speed-up results were obtained during model generation for yield estimation.

The ES algorithm was modified so as to use Metropolis criterion as the selection method. The modification involves the use of a pseudo-individual having average cost in Boltzmann trials. The resulting selection mechanism was compared with elitist selection. Elitist selection suffered from premature convergence while the proposed selection mechanism was able to converge to a much better solution. GA and ES also suffer from poor fine-tuning at the final stage of evolution. This drawback was eliminated with the selection mechanism and self-adaptation of mutation step-sizes. In addition to mutation step-sizes, recombination coefficients and cost function weights were also self-adapted. A method to prevent the search algorithm from cheating during self-adaptation of weights was proposed and shown to be effective. Addition of automatic determination routines for annealing parameters also minimized the search algorithm tuning effort necessary. Three synthesis examples, of different complexity and nature, were presented to validate the usefulness of SACSES.

A yield estimation mechanism based on piecewise cubic Hermite spline modeling of response surface was proposed. The estimation mechanism is faster than standard

simulation based yield estimation, and can be embedded in the search loop. A BTS OPAMP was synthesized using the yield-aware version of SACSES where Hermite model based yield estimation was introduced during the later stage of evolution. Errors due to the Hermite model were corrected during the final stages of evolution with the help of SPASE-based MC simulations. Parallelization methods were suggested in order to decrease the time required for yield-aware synthesis.

A hierarchical genetic algorithm structure was proposed for integrating the system and circuit levels. The method takes advantage of the fast initial convergence of the genetic algorithms. The proposed hierarchical scheme was applied to the synthesis of a 3^{rd} order Butterworth filter, with macro-model based and transistor-level simulation approaches. Results similar in performance, area and power to directly embedding the circuit simulator in the synthesis loop were obtained in a much shorter time. The second approach, in addition to providing a speedup of 15 times, eliminates the need for macro-model generation and performance equation derivation.

A feedback loop aimed to minimize the effects of inevitable layout parasitics was proposed and tested by synthesizing a BTS OPAMP circuit. It was observed that the feedback loop was able to account for the deviations caused by the layout parasitics. Two iterations through the loop were necessary in order to obtain a circuit within acceptable performance criteria.

8.1. Future Work

The proposed circuit level design automation tool, SACSES, and the design environment formed by the integration of the system, circuit and layout level design automation blocks have some properties that need to be improved. In order to enlighten the future work, the possible ways of improvement are described in the following paragraphs.

Although most of the search parameters of the proposed ES algorithm is determined automatically, there remains some parameters to be given by the user. The most important of these is the population size. Although convergence is possible with a conservatively large population size, as in Section 4.3, the synthesis time may not be optimal. For this reason, the population size must be adjusted according to the hardness of the problem. This adjustment can be made a-priori by simulating several random initial solutions and evaluating a hardness value based on the results. A neater approach is to vary the population size dynamically throughout the evolution process. There are several varying population size approaches. An age and lifetime value can be incorporated into each individual according to their performance as in GAVaPS [76], or the population size can be adapted depending on the ease or difficulty of the algorithm to generate offspring that outperform their parents, as in [77]. A survey of the adaptive population sizing schemes is given in [78]. These schemes will be inspected and those applicable to analog synthesis will be experimented to find the most suitable approaches.

The integration of the synthesis various design abstraction levels hierarchically is a new research topic. Some integration schemes were proposed in Chapter 6 and alternative schemes can be developed and experimented with different circuit topologies.

Simulation-based nature of SACSES allows easy addition of simulators other than SPASE. This maybe necessary for synthesizing some nonlinear or time-variant circuit topologies. For example, a harmonic balance simulator can be implemented and added to synthesize mixers. SACSES can also be integrated with an existing simulator like HSPICE by only altering the circuit evaluation function. In this case, acceleration mechanisms of SPASE cannot be utilized but the robust ES-based algorithm is still available. Also, integration of an aging simulator provides aging-aware synthesis.

Integration with the layout generator, TOLAS, will be experimented with different topologies after some aspects of TOLAS are improved. First, the need for user-given

layout templates contrasts the aim of minimizing initial human effort. An automated template generator, which is under development in another thesis study, is needed. Another point is better minimization of layout parasitics with the help of a sensitivity analyzer.

In Chapter 5, a parallelization scheme where simulations for both model development and performance evaluation are distributed was presented. This scheme will be experimented in order to assess its efficiency.

REFERENCES

1. Moore, G. E., "Cramming more components onto integrated circuits," *Electronics Magazine*, vol. 38, no. 8, April 1965.

2. Lopez, R. C., *Methodologies for Reusability and Design of Analog Integrated Circuits*, Ph.D. Thesis, Sevilla University, Spain, 2004.

3. Debyser, G., G. Van der Plas, F. Leyn, K. Lampaert, J. Vandenbussche, G. Gielen, W. Sansen, P. Veselinovic and D. Leenaerts, "AMGIE : A Synthesis Environment for CMOS Analog Integrated Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 20, No. 9, pp. 1037-1058, September 2001.

4. Gielen, G. G. E. and R. A. Rutenbar, "Computer-Aided Design of Analog and Mixed-Signal Integrated Circuits," *Proceedings of the IEEE*, Vol. 88, No. 12, pp. 1825-1852, December 2000.

5. Therasse, Y., L. Reynders, R. Lannoo, and B. Dupont, "A switched-capacitor filter compiler," *VLSI System Design*, vol. 8, no. 10, pp. 85-88, Sept 1987.

6. Assael, J., P. Senn, and M. S. Tawfik, "A switched-capacitor filter silicon compiler," *IEEE Journal of Solid-State Circuits*, vol. 23, pp. 166-174, Feb 1988.

7. Chang, H., E. Liu, R. Neff, E. Felt, E. Malavasi, E. Charbon, A. Sangiovanni-Vincentelli, and P. R. Gray, "Top-down, constraint-driven, methodology based generation of n-bit interpolative current source D/A converters," *Proceedings of IEEE CICC*, pp.369-372, May 1994.

8. Azeri, O. D., *Analog Circuit Optimization with Hierarchical Genetic Algorithms - 3^{rd} Order Low-pass Butterworth Filter Example*, MS Thesis, Boğaziçi University, İstanbul, 2009.

9. Harjani, R., R.Rutenbar, and L. R. Carley, "OASYS: A framework for analog circuit synthesis," *IEEE Trans. Computer-Aided Design*, vol.8, pp. 1247-1265, Dec. 1989.

10. El-Turky, F. and E. Perry, "BLADES: An artificial intelligence approach to analog circuit design," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 680-692, June 1989.

11. Sheu, B. J., A. H. Fung and Y. N. Lai, "A knowledge -based approach to analog IC design," *IEEE Trans. Circuits and Systems*, vol. 35, pp. 256 - 258, 1988.

12. Degrauwe, M. et al., "IDAC:An interactive design tool for analog CMOS circuits," *IEEE J. Solid-State Circuits*, vol. SC-22, pp. 1106-1116, Dec. 1987.

13. Massier, T., H. Graeb, and U. Schlichtmann, "The sizing rules method for CMOS and bipolar analog integrated circuit synthesis," *IEEE Trans. Computer-Aided Design*, vol. 27, pp. 2209-2222, Dec. 2008.

14. Somani, A., P. P. Chakrabarti, and A. Patra, "An evolutionary algorithm-based approach to automated design of analog and RF circuits using adaptive normalized cost functions," *IEEE Trans. Evolutionary Computation*, vol. 1, pp. 336-353, June 2007.

15. Koh, H. Y., C. H. Sequin, and P. R. Gray, "OPASYN: A compiler for CMOS operational amplifiers," *IEEE Trans. Computer-Aided Design*, vol.9, pp. 113-125, Feb. 1990.

16. Gielen, G., H. Walscharts, and W. Sansen, "Analog circuit design optimization based on symbolic simulation and simulated annealing," *IEEE J. Solid-State Circuits*, vol. 25, pp. 707-713, June 1990.

17. Hjalmarson, E., *A computer-aided approach to design of robust analog circuits*, Ph.D. Thesis, Inst. of Tech., Linkping Univ., Linkping, 2006.

18. Vladimirescu, A. and R. Zlatanovici, "Analog circuit synthesis using standard EDA tools," *in Proc. IEEE International Symp. on Circuits and Systems, ISCAS 2006*, pp.5239-5242, May 2006.

19. Papadopoulos, S., R. J. Mack, and R. E. Massara, "A hybrid genetic algorithm method for optimizing analog circuits," *in Proc. 43rd IEEE Midwest Symp. on Circuits and Systems*, Lansing MI, pp.140-143, Aug 2000.

20. Yuan, J., N. Farhat, and J. V. der Spiegel, "GBOPCAD: A synthesis tool for high performance gain-boosted opamp design," *IEEE Trans. Circuits Syst. I, Fundamental Theory and Applications*, vol.52, pp. 1535-1544, Aug. 2005.

21. Krasnicki, M., R. Phelps, R. A. Rutenbar, and L. R. Carley, "MAEL-STROM: Efficient simulation-based synthesis for custom analog cells," *in Proc. IEEE/ACM Design Automation Conf.*, pp. 951-957, June 1999.

22. Phelps, R., M. Krasnicki, R. A. Rutenbar, L. R. Carley, and J. R. Hellums, "Anaconda: Simulation-based synthesis of analog circuits via stochastic pattern search," *IEEE Trans. Computer-Aided Design*, vol. 19, 703-717, June 2000.

23. Ochotta, E., R. Rutenbar, and L. R. Carley, "Synthesis of high-performance analog circuits in ASTRX/OBLX," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 273-294, Mar. 1996.

24. Nye, W., D. C. Riley, A. Sangiovanni-Vincentelli, and A. L. Tits, "DELIGHT.SPICE: An optimization-based system for the design of integrated circuits," *IEEE Trans. Computer-Aided Design*, vol. 7, pp. 501-519, Apr. 1988.

25. Bayraktaroğlu, İ., *Circuit level simulation based training algorithms for analog neural networks*, MS thesis, Dept. of Electrical and Electronics Eng., Boğaziçi University, İstanbul, 1996.

26. Sönmez, Ö. S., *An optimization-based hierarchical analog design automation sys-*

tem, MS Thesis, Dept. of Electrical and Electronics Eng., Boğaziçi University, İstanbul, 2003.

27. Mandal, P. and V. Visvanathan, "CMOS op-amp sizing using a geometric programming formulation," *IEEE Trans. Computer-Aided Design*, vol. 20, pp. 22-38, Jan. 2001.

28. Hershenson, M., S. P. Boyd, and T. H. Lee, "Optimal design of a CMOS opamp via geometric programming," *IEEE Trans. Computer-Aided Design*, vol. 20, pp. 1-21, Jan. 2001.

29. Dawson, J. L., S. P. Boyd, M. del Mar Hershenson and T. H. Lee, "Optimal allocation of local feedback in multistage amplifiers via geometric programming," *IEEE Trans. Circuits Syst. I, Fundamental Theory and Applications*, vol. 48, pp. 1-11, Jan. 2001.

30. Hersherson, M. d. M., "Design of pipeline analog-to-digital converters via geometric programming," *in IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 317-324, San Jose, California, Nov. 2002.

31. Xu, Y., K. L. Hsiung, X. Li, L. T. Pileggi and S. P. Boyd, "Regular analog/RF integrated circuits design using optimization with recourse including ellipsoidal uncertainty," *IEEE Trans. Computer-Aided Design*, vol. 28, pp. 623-637, May 2009.

32. Li, X., P. Gopalakrishnan, Y. Xu and L. T. Pileggi, "Robust analog/RF circuit design with projection-based performance modeling," *IEEE Trans. Computer-Aided Design*, vol. 26, pp. 2-15, Jan. 2007.

33. Wolfe, G. and R. Vemuri, "Extraction and use of neural network models in automated synthesis of operational amplifiers," *IEEE Trans. Computer-Aided Design*, vol. 22, pp. 198-212, Feb. 2003.

34. Stehr, G., H. E. Graeb, and K. J. Antreich, "Analog performance space exploration by normal-boundary intersection and by Fourier-Motzkin elimination," *IEEE Trans. Computer-Aided Design*, vol. 26, 1733-1748, Oct. 2007.

35. Ingber, L., "Very fast simulated re-annealing," *Mathl. Comput. Modelling*, vol. 12, pp. 967-973, 1989.

36. Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, 1998.

37. Eiben, A. E. and J. E. Smith, *Introduction to Evolutionary Computing*, Springer, 2003.

38. Mahfoud, S. W. and D.E. Goldberg, "Parallel Recombinative Simulated Annealing: A Genetic Algorithm," *Parallel Computing*, vol. 21, pp. 1-28, 1995.

39. Ramos, J., K. Francken, G. G. E. Gielen and M. S. J. Steyaert, "An efficient, fully parasitic-aware power amplifier design optimization tool," *IEEE Trans. Circuits Syst. I, Fundamental Theory and Applications*, vol.52, pp. 1526-1534, Aug. 2005.

40. Liu, B., et al., "Analog circuit optimization system based on hybrid evolutionary algorithms," *Integration, the VLSI Journal*, vol. 42, issue 2, pp. 137-148, Feb. 2009.

41. Park, J. and D. Allstot, "RF circuit synthesis using particle swarm optimization," *in Proc. IEEE International Symp. on Circuits and Systems, ISCAS 2004*, vol. 5, pp. V-93-V-96, May 2004.

42. Ceperic, V., Z. Butkovic and A. Baric, "Design and optimization of self-biased complementary folded cascode," *in IEEE Mediterranean Electrotechnical Conference, MELECON*, pp. 145-148, Benalmadena, Spain, May 2006.

43. Silva, J. and N. Horta, "GENOM: Circuit-level optimizer based on a modified genetic algorithm kernel," *in Proc. IEEE International Symp. on Circuits and*

Systems, ISCAS 2002, vol. 1, pp. I-745-I-748, May 2002.

44. Barros, M., J. Guilherme and N. Horta, "An evolutionary optimization kernel using a dynamic GA-SVM model applied to analog IC design," *18th European Conference on Circuit Theory and Design, ECCTD 2007*, pp. 32-35, Aug 2007.

45. Cho, H. J., S. Y. Oh and D. H. Choi, "A new evolutionary programming approach based on simulated annealing with local cooling schedule," *in IEEE World Congress on Computational Intelligence*, pp. 598-602, May 1998.

46. Shichman, H. and D. A. Hodges, "Modeling and simulation of insulated-gate field-effect transistor switching circuits," *IEEE J. Solid-State Circuits*, vol. 3, no. 5, pp. 285-289, Sep. 1968.

47. Foty, D. P., *MOSFET Modeling with SPICE - Principles and Practice*, Prentice Hall, Upper Saddle River, NJ, 1997.

48. Sheu, B. J., D. L. Scharfetter, P. Ko, and M. Jen, BSIM: Berkeley Short-Channel IGFET Model for MOS transistors, *IEEE J. Solid-State Circuits*, vol. 22, no. 4, pp. 558-566, Aug. 1987.

49. Shichman, H. and D. A. Hodges, "Modeling and simulation of insulated-gate field-effect transistor switching circuits," *IEEE J. Solid-State Circuits*, vol. 3, no. 5, pp. 285-289, Sep. 1968.

50. Liu, W., *MOSFET Models for SPICE Simulation, Including BSIM3v3 and BSIM4*, John Wiley & Sons, New York, 2001.

51. Enz, C. C., F. Krummenacher, and E. A. Vittoz, "An analytical MOS transistor model valid in all regions of operation and dedicated to low-voltage and low-current applications," *J. Analog Integr. Circuits Signal Process.*, vol. 8, pp. 83-114, July 1995.

52. Alpaydın, G., *A new approach to analog integrated circuit optimization*, Ph.D.

Thesis, Dept. of Electrical and Electronics Eng., Boğaziçi University, İstanbul, 2000.

53. Brent, R. P., *Algorithms for Minimization Without Derivatives*, Prentice-Hall, 1973.

54. Malik, N. R., *Electronic Circuits, Analysis, Simulation, and Design*Prentice Hall, 1995.

55. Chong, Z. Y. and W. M. C. Sansen, *Low-Noise Wideband Amplifiers in Bipolar and CMOS Technologies*, Norwell, MA: Kluwer Academic, 1990.

56. Razavi, B., *Design of Analog CMOS Integrated Circuits*, McGraw Hill, 2001.

57. Xie, D., M. Cheng and L. Forbes, "SPICE models for flicker noise in n-MOSFETs from subthreshold to strong inversion," *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 1293-1303, Nov. 2000.

58. Vlach, J. and K. Singhal, *Computer Methods for Circuit Analysis and Design*, Van Nostrand Reinhold, 1983.

59. Talay, S., E. Deniz and G. Dündar, "A sigma-delta ADC design automation tool with embedded performance estimator," *Integration, the VLSI Journal*, vol. 42, issue 2, pp. 181-192, Feb. 2009.

60. Alpaydın, G., S. Balkır and G. Dündar, "An evolutionary approach to automatic synthesis of high-performance analog integrated circuits," *IEEE Trans. Evolutionary Computation*, vol. 7, pp. 240-252, Jun. 2003.

61. Kirkpatrick, S., "Optimization by simulated annealing: quantitative studies" *Journal of Statistical Physics*, vol. 34, pp. 975-986, March 1984.

62. Kirkpatrick, S., C.D. Gerlatt, and M.P. Vecchi, "Optimization by simulated annealing", *Science*, vol. 220, pp. 671-680, May 1983.

63. Sanchez, H. S. and J. F. Solis, "A methodology to parallel the temperature cycle in simulated annealing," *in Lecture Notes in Computer Science, MICAI 2000: Advances in Artificial Intelligence*, vol. 1793, pp. 63-74, 2000.

64. Eiben, A. E., B. Jansen, Z. Michalewicz, B. Paechter, "Solving CSPs using self-adaptive constraint weights: how to prevent EAs from cheating," *in The Genetic and Evolutionary Computation Conference, GECCO 2000*, pp. 128-134, 2000.

65. Haspeslagh, J. and W. Sansen, "Design techniques for fully differential amplifiers", *in IEEE Custom Integrated Circuits Conference, CICC 1988*, pp. 12.2.1-12.2.4, 1988.

66. Wang, Z. and S. Director, "An efficient yield optimization method using a two-step linear approximation of circuit performance," *in Proc. IEEE European Design and Test Conference*, pp. 567-571, 1994.

67. Schenkel, F., M. Pronath, S. Zizala, R. Schwencker, H. Graeb and K. Antreich, "Mismatch analysis and direct yield optmization by spec-wise linearization and feasibility-guided search," *in Proc. IEEE Design Automation Conference*, pp. 858-863, 2001.

68. Li, X., Y. Zhan and L. T. Pileggi, "Quadratic statistical MAX approximation for parametric yield estimation of analog/RF integrated circuits," *IEEE Trans. Computer-Aided Design*, vol. 27, pp. 831-843, May 2008.

69. Li, X., J. Le, P. Gopalakrishnan and L. T. Pileggi, "Asymptotic probability extraction for nonnormal performance distributions," *IEEE Trans. Computer-Aided Design*, vol. 26, pp. 831-843, Jan 2007.

70. Pelgrom, M.J.M., AAD C. J. Duinmaijer and A. P. G. Welbers, "Matching properties of MOS transistors," *IEEE Journal of Solid-State Circuits*, vol. 24, pp. 1433-1440, October 1989.

71. Conte, S. and C. deBoor, *Elementary Numerical Analysis*, McGraw Hill, 1972.

72. Luchetta, A., S. Manetti and A. Reatti, "SAPWIN-a symbolic simulator as a support in electrical engineering education," *IEEE Trans. on Education*, vol.44, no.2, pp. 213-222, May 2001

73. Unutulmaz, A., *Analog Layout Synthesizer for a Parasitic-aware Design Loop*, M.S. Thesis, Boğaziçi University, İstanbul, 2009.

74. Ataç, M. S., *A Flexible and High Performance Simulation Based Sensitivity Analysis Tool For Analog Layout Constraint Generation*, M.S. Thesis, Boğaziçi University, İstanbul, 2003.

75. Şen, T., *Yet another simulation based sensitivity analysis tool for analog layout generation*, M.S. Thesis, Boğaziçi University, İstanbul, 2007.

76. Arabas, J., Z. Michalewicz, and J. Mulawka, "GAVaPS a genetic algorithm with varying population size," *In Proc. of the First IEEE Conf. on Evolutionary Computation*, pp. 7378, Piscataway, NJ, 1994.

77. Affenzeller, M., S. Wagner, and S. Winkler, "Self-adaptive Population Size Adjustment for Genetic Algorithms," *EUROCAST 2007*, pp. 820-828, 2007.

78. Lobo, F. G. and C. F. Lima, "Adaptive Population Sizing Schemes in Genetic Algorithms," *Studies in Computational Intelligence (SCI) 54*, pp. 185-204, 2007.