I, Huiying Yang, hereby submit this work as part of the requirements for the degree of:

PHD

in:

Computer Engineering

It is entitled:

Symbolic Sensitivity Analysis Techniques and Applications in Analog Circuit Synthesis

This work and its defense approved by:

Chair: Dr. Ranga Vemuri
Dr. Hal Carter
Dr. Wen-Ben Jone
Dr. Karen Tomko
Dr. Jintai Ding
Symbolic Sensitivity Analysis Techniques and
Applications in Analog Circuit Synthesis

A dissertation submitted to the
Division of Research and Advanced Studies
of the University of Cincinnati
in partial fulfillment of the
requirements for the degree of
DOCTOR OF PHILOSOPHY
in the Department of
Electrical and Computer Engineering and Computer Science
of the College of Engineering
by
Huiying Yang
Bachelor of Engineering, Beijing University of Aeronautics and Astronautics, 1994
Thesis Advisor and Committee Chair: Dr. Ranga Vemuri
December 13, 2006
Abstract

With increasing complexity of integrated circuits and decreasing time-to-market constraints, there was remarkable success in automation of the digital circuit design, but it exposes the lack of comparable analog design automation tools. Although analog components typically constitute only a small portion of the integrated circuits and emerging systems-on-chip (SoC) designs, they are indispensable in all electronic applications that interface with the outside world. Due to the complexity of analog and radio-frequency (RF) circuits, analog circuit design time dominates and becomes a bottleneck in ICs.

There is a growing need for analog design automation tools. One of the main challenges in analog design is sizing. Given an analog topology and a set of high-level design specifications, the size of all components in the circuit must be appropriately determined in order to meet all design target specifications. There are many ways to approach the sizing problem, from manual design and knowledge-based strategies to global optimization engines utilizing a detailed circuit simulator like SPICE.

Typically, most commercial approaches use simulation-in-loop to tackle analog sizing problem due to the high accuracy desired from the estimation process which introduce commercial simulators, like SPICE, to numerically estimate circuit performance and identify critical parameters. Choosing appropriate parameter for sizing in each loop is very critical during the whole sizing process. Sensitivity of an analog circuit is the mathematical measure of variations in the performance metrics due to infinitesimally small perturbations of circuit parameter values which can determine the critical design variables in analog circuit synthesis and can be used repetitively to identify critical parasitics that severely hamper the circuit performance.
In this dissertation, we focus on the development of accurate and efficient symbolic sensitivity analysis for use in the synthesis of analog circuits. We build the sensitivity analysis model and propose the methodologies for performance sensitivity calculations with respect to transistors, capacitors and MOSFETs. We apply different sensitivity analysis techniques to improve the efficiency and accuracy of analog circuit synthesis. In our approach, the parameters chosen for sizing are determined by their sensitivities instead of randomly choosing the parameters to be perturbed. This reduces the iteration time and hence the overall synthesis time considerably. We also build the parasitic estimation model and optimize the parasitics by applying symbolic sensitivity techniques.
Acknowledgements

I would like to express my deepest gratitude and sincerest thanks to Dr. Ranga Vemuri for his constant guidance and support. Dr. Vemuri’s suggestions and recommendations have been invaluable for the successful completion of my thesis. I would like to thank Dr. Hal Carter, Dr. Wen-Ben Jone, Dr. Karen Tomko and Dr. Jintai Ding for being a part of my dissertation committee and offering me valuable suggestions.

This work was supported in part by the Defense Advanced Research Projects Agency and the Sensors Directorate of the Air Force Research Laboratory, U.S. Air Force, Wright Patterson AFB, OH, under Contract F33615-01-C-1977, and the National Science Foundation under award number CCF-0429717. I would like to acknowledge their support.

I would like to thank Dr. Georges Gielen and Wim Verhaegen, Katholieke Universiteit, Leuven, Belgium, for their permission to use the DDKUL software. Their guidance also has been very important in my research. I thank Dr. Andrea Pacelli for the VPEC software suite.

I also thank my colleagues Anuradha and Veena Yelamanchili for letting me use their software. In particular, I thank Mukesh Ranjan for his persistent assistantance and support on my research.

Finally, I owe a big thanks to all my friends and labmates for their wonderful company. Thank you all.
# Contents

1 Introduction ................................................................. 1
   1.1 Digital and Analog Circuit Synthesis ................................. 2
   1.2 Symbolic Analysis of Analog Circuit Synthesis ......................... 6
   1.3 Circuit Sizing ........................................................... 8
   1.4 Layout-Aware Analog Circuit Synthesis ................................. 8
      1.4.1 Parasitic-aware circuit synthesis .................................. 10
      1.4.2 Layout-inclusive circuit synthesis .................................. 11
   1.5 Research Approaches .................................................... 12
      1.5.1 Efficient sensitivity analysis based on ECDs ....................... 13

2 Sensitivity Analysis Overview ............................................. 16
   2.1 Perturbation Method ...................................................... 17
   2.2 Direct Method ............................................................ 18
   2.3 Adjoint-based Sensitivity Approach .................................... 19

3 Sensitivity Analysis Based on ECDs ...................................... 21
   3.1 Introduction ............................................................. 21
   3.2 Overview of Determinant Decision Diagrams (DDDs) and Element-Coefficient Diagrams (ECDs) ................................................. 23
6 Efficient Symbolic Sensitivity based Parasitic-Inclusive Optimization in Layout-Aware Analog Circuit Synthesis

6.1 Introduction .................................. 70
6.2 Parasitic Modeling .................................. 73
   6.2.1 Physical layout extraction technique .................... 73
   6.2.2 Parasitic model estimation technique ..................... 74
6.3 Symbolic Sensitivity Based Parasitic Optimization ................. 75
6.4 Analog Circuit Synthesis Implementation With Proposed Approach ........ 77
6.5 Experiments ........................................ 78
6.6 Conclusions ....................................... 81

7 Conclusions and Future Work .................................. 85

7.1 Contributions ........................................ 85
7.2 Future Directions ........................................ 87
7.3 Summary ............................................... 91
## List of Figures

1.1 Digital Synthesis Flow ........................................ 4
1.2 The Two Basic Approaches Towards Analog Circuit Synthesis .......... 6
1.3 Optimization-based Circuit Sizing Flow ............................. 9
1.4 Optimization-based Circuit Sizing Flow ............................. 11
1.5 Optimization-based Circuit Sizing Flow ............................. 12
1.6 Framework of the Proposed Approaches ............................. 14

3.1 An Illustrative RC Network ........................................ 24
3.2 DDD Representation of Vout for RC Network ......................... 25
3.3 ECD Representation of Vout for RC Network ......................... 25
3.4 Basic Connection Types ............................................. 26
3.5 ECD Representation of Derivative of $I_n$: (Left) When $c = p_j$ (Right) When $c \neq p_j$ . . . 29
3.6 Benchmark Circuits (a) Single Transistor Amplifier (b) Single-Ended LNA . . . 32
3.7 Two Stage Opamp1 .................................................. 32
3.8 Speedup of Four Benchmark Circuits under Different Numbers of Frequency in One Iteration ................................ 37

4.1 Proposed Analog Circuit Synthesis Approach ........................ 41
4.2 Basic MOS Transistor Model ....................................... 44
List of Tables

3.1 Maximum Difference in Gain Sensitivity between SSAE and SPECTRE (for RC and STA) ........................................ 33
3.2 Maximum Difference in Gain Sensitivity between SSAE and SPECTRE (for TSO and SE-LNA ) ..................................... 33
3.3 TSO Gain Transfer Function Sensitivities w.r.t. C1 for Different Values ............ 34
3.4 SE-LNA Transfer Function Sensitivity w.r.t. C1 for Different Frequencies ........... 34
3.5 Time Results for Sensitivity Analysis of Gain for 500 Iterations .................... 35
4.1 Synthesis Result Comparison without/with Multiparameter Sensitivity Analysis ... 52
5.1 ECD Sizes of Benchmark Circuits ........................................ 68
5.2 Iteration and Runtime Comparison without/with Sensitivity Weight Optimization Algorithm ........................................ 68
5.3 Iteration and Runtime Comparison without/with Same Sensitivity Algorithm Weight 69
6.1 Parasitic Capacitances of Benchmark Circuits ....................................... 82
6.2 Layout-Aware Circuit Performance Comparison ....................................... 83
6.3 Iteration and Runtime Comparison Using Parasitic Modeling between with and without Optimized Parasitics Method ........................................ 84
Chapter 1

Introduction

In recent years, with the demand for decreasing device sizes of the integrated circuits, the complete systems have been integrated on a few chips or even on one single chip which might occupy one or more boards previously. Analog and digital circuits are also being integrated on the same die for reasons of reducing cost and improving performance. Despite the trend previously to replace analog circuit functions with digital computations, there are some typical functions that will always remain analog due to the need of the interface between the digital world and the real world such as in communications and signal processing fields. Most signals of practical interest, such as speech, biological signals, seismic signals, radar signals, and various communications signals such as audio and video signals, are analog. Also the largest analog circuits today are high performance (high-speed, low-power) digital circuits.

In the digital domain, design automation tools are well developed and commercially available today. However, analog design automation is not comparable with its digital counterpart. Analog design is a very complex and knowledge-intensive process which needs manual and iterative design methodologies and heavily relies on circuit understanding and related design heuristics. The need to develop efficient analog design automation tools has become imperative in order to keep the pace
with digital design automation to reduce the whole design cycle and hence the time-to-market.

1.1 Digital and Analog Circuit Synthesis

Circuit synthesis (also called schematic synthesis) is the process of determining from the high-level, abstract specifications to the low-level details to achieve those specifications. Circuit synthesis creates a circuit topology, the correct sizing and biasing of the devices in that topology to meet input performance specifications. Circuit synthesis is the inverse operation of circuit analysis, where the subblock parameters (such as device sizes and bias values) are given and the resulting performance of the overall block is calculated (as is done in SPICE). During synthesis, the block performance is specified and values for the subblock parameters needed to meet these specifications have to be determined. This inverse process is not a one-to-one mapping, but usually is an underconstrained problem with many degrees of freedom.

In a digital synthesis setting, hardware description languages such as VHDL or Verilog have been successfully used to represent digital systems at both behavioral and structural levels. Because of the maturity and stability of digital circuit synthesis, single chip designs exceeding 100 million transistors are common and relatively inexpensive. Digital synthesis is a translation process which takes a design description from one level of abstraction to a lower level. Synthesis is essentially a behavior to structure translation. The overall translation process is similar to the compilation of a high-level programming language representation into executable code. The inputs to a synthesis tool usually include a HDL description (timing, area goals and a technology library). The final outputs are optimized netlists, estimated performance, and the ‘functional’ area part of the end design to be completed through layout. This process will also determine the ‘interconnect’ part of the total area. The entire stepwise synthesis process is typically subdivided into the following four steps:
Behavioral Synthesis  Behavioral synthesis is a process of translating, say a C-like, algorithmic description (HDL) into a register-transfer-level (RTL) structure. This entails determining the number of clock cycles and separating out datapaths, memories, and control units ruling the same datapaths and memories. Although the exact nature of datapath operators may be left to ensuring synthesis step, the datapath architecture (e.g. number of datapath registers, pipeline depths etc.) becomes fixed in this step. The result (output) is normally written out as RTL code so superficially the process appears to be a behavior to behavior translation.

RTL Synthesis  RTL synthesis generates structural netlists from a set of register transfer functions delivered by the behavioral synthesis step. The state boundary (clock by clock behavior) is already defined at this level. The register transfer operations can be described as finite state machines and transfer level equations. The tasks include state minimization, state encoding, logic minimization and technology mapping.

Logic Synthesis  Logic synthesis translates Boolean expressions into digital logic gates. This transformation is required because the functional units used in the RTL design may not have a direct correspondence with the components available in the library. First the design is flattened into a collection of gates. Then these generic gates are mapped to library specific gates by a technology mapping phase. The optimization of logic synthesis is usually divided into two phases: a front end phase which minimizes the circuit independent of the technology library, and a back end phase which maps a structural netlist to an interconnection of library cells.

Physical Synthesis  During physical synthesis the gate level netlist is finally placed and routed to obtain a layout for the system which has to be realized on a chip that is essentially two-dimensional. Then the interconnections among the gates are realized by the routing phase that uses wires to re-
alize the connections.

A typical digital synthesis flow has been represented in Figure 1.1.

![Digital Synthesis Flow Diagram](image)

**Figure 1.1: Digital Synthesis Flow**

While the analog part of a mixed signal system is usually much smaller than the digital part the design time for the analog circuitry often either equals or exceeds the digital part which is often the bottleneck and dominates the design cycle. There are two basic analog synthesis approaches shown in Figure 1.2. The first class of analog synthesis systems presented in the mid to late eighties was knowledge-based. Specific heuristic design knowledge about the circuit topology under design was encoded explicitly in some computer executable form which was then executed during the synthesis run for a given set of input specifications to obtain the design solution. The knowledge was encoded in different ways in different systems. The design equations specific for a particular circuit topology had to be derived and the degrees of freedom in the design had to be solved explicitly during the development of the design plan using simplifications and design heuristics.
The big advantage of using design plans is their fast execution speed, which allows for fast performance space explorations. The big disadvantages are the lack of flexibility and the large time needed to develop a plan for each topology and design target as analog design heuristics are very difficult to formalize in a general and context-independent way. Considering the large number of circuit schematics in use in industrial practice, this essentially restricted the commercial usability of the tool and limited its capabilities to the initial set of schematics delivered by the tool developer [4].

In order to make analog design systems much more open for new circuit schematics, an alternative solution was developed since the late eighties by using optimization techniques to implicitly solve the degree of freedoms in analog design while optimizing the performance of the circuit under the given specification constraints. The approach is illustrated schematically in Figure 1.2. At each iteration of the optimization procedure the performance of the circuit has to be evaluated. There are two kinds of performance evaluation methods as shown in the right side of Figure 1.2. One is based on commercial simulator, like SPICE, the other is based on symbolic models. Many of the current commercially available synthesis tools use SPICE in the loop to actively provide performance attributes of sized circuits. Simulator-based approach is applicable to an arbitrary circuit and mostly trustworthy to designers. On the other hand, this approach is computationally expensive due to the requirement of the numerical circuit simulation. Therefore, researches on this area have focused on the reduction of computational cost based on conventional approaches such as task parallelization, the formulation of a proper cost function and high convergence rate of optimization algorithms.

One method of dealing with the problem of long simulation times involves constructing accurate and efficient models for symbolic performance evaluation. Once constructed these models stand in for the prohibitively expensive SPICE simulations during circuit synthesis. These models
may be reused many times to efficiently synthesize the topology. The other method is to introduce sensitivity analysis to increase high convergence rate of synthesis so as to reduce total synthesis loop hence the runtime. This dissertation discusses methods for the construction of accurate and efficient symbolic performance model and analysis their performance sensitivities with respect to the circuit parameters.

![Diagram](image)

Figure 1.2: The Two Basic Approaches Towards Analog Circuit Synthesis

### 1.2 Symbolic Analysis of Analog Circuit Synthesis

Symbolic analysis at the circuit level is a formal technique to calculate the behavior or a characteristic of a circuit with the independent variable. The technique is complementary to numerical analysis (where the variables and the circuit elements are represented by numbers) and qualitative analysis (where only qualitative values are used for voltages and currents, such as increase, decrease or no change) [14].
Symbolic circuit analysis techniques have been developed to help designers gain a better understanding of a circuit’s behavior. A symbolic simulator is a computer tool that takes the input as an ordinary (SPICE-type) netlist and returns as output (simplified) analytic expressions for the requested circuit network functions in terms of the symbolic representations of the frequency variable and (some or all of) the circuit elements. They perform the same function as the designers traditionally do by hand analysis. The difference is that the analysis is now much faster done by the computer, which can handle more complex circuits and does not make as many errors [13]. Symbolic analysis can also be used for behavioral model generation. Another use is in testing and fault diagnosis of analog circuits. Symbolic analysis techniques can also be used for poles-zeros extraction and performance model analysis.

Compared with numerical analyzers (like SPICE), symbolic analyzers has several incomparable advantages. The equations generated by symbolic analysis help designers gain a better understanding of circuit behavior. In this way, the symbolic equations can help not only understand the first-order functional behavioral of an analog circuit, but also provide insight into second-order effects in a circuit which is equally important for the correct functioning of the design in its system application later on. Whereas, numerical analysis results are just numbers which provide no insight at all. When you do a simulation, numerical analysis has to be repeated again and again if the parameters vary, which wastes a lot of time. However, symbolic equations remain valid for different values of parameters.

Although the symbolic analyzer has these advantages symbolic analysis are only limited in linear or weakly nonlinear AC analysis (small-signal linearized circuits) and so it’s application is not very popular. Also since symbolic analyzer generates large, expanded expressions, which restrict its application in large circuits. However it’s tackled by using complicated hierarchical decomposition method. Symbolic techniques are still on its way. We can’t completely substitute numerical
analysis as a means of circuit analysis, but we still can take best use of symbolic techniques in some applications.

In this dissertation we use symbolic performance evaluation instead of commercial simulators (like SPICE) in order to reduce the performance evaluation time which dominates the synthesis cycle. We also propose symbolic sensitivity analysis to determine the critical parameters in circuit sizing process which decreases the iterations considerably.

1.3 Circuit Sizing

Analog synthesis is complicated because it not only consists of topology and layout synthesis but also consists of component sizing, which is a crucial step in analog circuit synthesis. Circuit sizing is the optimization process to search for a design parameter set \( x \) that makes a circuit meet target specifications. Generally, design parameter set \( x \) includes transistor dimensions, passive component values and bias currents in transistor-level circuit sizing. Similar to the synthesis categories described above, circuit sizing falls into two categories. One is knowledge-based circuit sizing; the other is optimization-based circuit sizing, which became the standard in recent years. Figure 1.3 presents a typical optimization-based circuit sizing process.

In this dissertation, we propose to reduce the time spent in circuit sizing by replacing the simulator in the optimization loop with fast and accurate symbolic performance models and using sensitivity analysis inclusive techniques to increase convergence rate during the synthesis loop.

1.4 Layout-Aware Analog Circuit Synthesis

The behavior of analog circuits is extremely sensitive to the parasitic effects which are introduced during the subsequent layout phase. Parasitics not only influence the circuit performance but also
lead to fail to meet the desired performance specifications, during post-layout verification. Especially for RF circuits, at GHz frequencies, all the parasitic elements of both the active and passive elements also come into play. This problem has raised the demand for layout-aware analog circuit synthesis methods. Layout-aware synthesis approach is that the parasitics are extracted or estimated in each synthesis iteration. After the optimization engine proposes a new set of values for circuit parameters, and before performances are estimated, it is imperative to analyze the correlation between the circuit parameter values and the effects due to their physical implementation. After each iteration, the new parasitics are introduced and the new numerical values of these parasitic components are updated. There are two approaches to solve layout-aware circuit synthesis process. The first approach is including parasitic estimation model in each iteration which is called parasitic-aware circuit synthesis. The other technique involves generating a complete physical layout in each iteration and the method is known as layout-inclusive circuit synthesis. This approach
is also termed as the layout-in-loop approach.

### 1.4.1 Parasitic-aware circuit synthesis

The problem of performance degradation due to layout parasitics can be alleviated by modeling the layout effects during the circuit design or synthesis which increases the total synthesis time due to the large number of parasitics included in the synthesis loop. When parasitic effects are considered, the complete circuit usually becomes much too complicated for hand analysis so finding the optimum solution is nearly impossible. Designers frequently have to decide among a large number of possible solutions representing different trade-offs. This motivates the essential need for parasitic optimization. One way to estimate the parasitics is to use mathematical expressions, which correlate the size of a module, with various parasitics associated with that particular module. The expresses can be obtained by a simple analysis of the layout-template of the module. The other popular technique is a table look-up method, where by using interpolation techniques, the parasitic values are guessed from pregenerated data. Both these method serve the purpose well for intra-module parasitics. The problem arises while estimating the interconnect parasitics. Due to the varying structure of interconnects, it is difficult, and often impossible, to accurately estimate the parasitics. Therefore, for a high-performance circuit, the performance of which is extremely sensitive to interconnect parasitics, the parasitic-estimation technique is inadequate, due to which the parasitic-aware synthesis method can not be used [27].

The parasitic-aware method is shown in Figure 1.4. This method builds parasitic macro-models by analyzing several instances of the layouts. These macro-models are used in the synthesis loop to yield fast and accurate parasitic estimates. Parasitic optimization is done outside the synthesis loop which can save significant runtime. This flow is capable of achieving designs with layout closure.
1.4.2 Layout-inclusive circuit synthesis

Typically, the analog layout-inclusive synthesis task is split up in analog sizing and analog layout generation processes. The sizing and layout generation are merged into one optimization process, effectively removing the redesign loop from layout to sizing that exists in the traditional analog synthesis approach.

The layout-inclusive synthesis method is shown in Figure 1.5. Circuit topology and search range for each independent circuit parameter are inputs to the optimization engine. The output is a set of values for each parameter in each iteration. Using this set of parameter values, either a layout is generated completely or it is just instantiated. The synthesis result is a physical layout, which is then extracted and verified to see if it meets the design requirements.

In this method, physical layout is generated to verify and check whether desired performance
Figure 1.5: Optimization-based Circuit Sizing Flow

specifications have been met after the layout generation and extraction. There steps repeated till the desired performance specifications are met. This approach is time-consuming and computationally expensive.

1.5 Research Approaches

Currently the optimization-based simulator-in-loop approach is the most popular and widely accepted. Analog synthesis tools for performance simulation are time consuming during circuit sizing. Two main specifications to judge the synthesis tools are time spent in circuit synthesis and the convergence rate of the procedure. Circuit simulation and performance evaluation dominate the overall circuit synthesis time.

We propose to devise the synthesis methodologies which are faster than simulation-in-loop
synthesis approaches and also speedup the convergence rate considerably.

The framework of the proposed approaches is shown in Figure 1.6. Our approach is presented with the shadow in the figure. Circuit topology and search range for each parameter value of the circuit are the inputs to the optimization engine. In this case new simulated annealing algorithm is used for optimization which is based on symbolic sensitivity weight analysis. A new set of parameter values are passed by the optimization engine to NGSPICE (or other simulators) to perform operating point (OP) analysis. The small-signal values for all active devices generated by OP analysis and values of capacitors and resistors are passed to the performance evaluator and sensitivity evaluator. For sensitivity analysis there are two methodologies: one is regular sensitivity analysis for capacitors and resistors; the other is multiparameter sensitivity analysis for MOSFETs. The performance evaluator processes the simulation data to obtain the values of the desired performance parameters. These parameters are passed to the cost function evaluator to evaluate the cost which is used by the optimization engine to judge if the specifications are met. Meanwhile, the sensitivity evaluator calculates the performance sensitivity values with respect to all active and passive devices, which are used to determine which parameter is chosen to be perturbed. This process is repeated until the desired performance goals are achieved.

The details of our approach will be described in the following sections.

1.5.1 Efficient sensitivity analysis based on ECDs

Sensitivity analysis plays an important role in determining the critical design variables in analog circuit synthesis. In layout-aware circuit sizing sensitivity analysis can be used repetitively to identify critical parasitics that severely hamper the circuit performance.

In this dissertation The remaining work has been organized as follows. Chapter 2 surveys three popular methods for sensitivity analysis in the circuits. The advantages and disadvantages of each
method are mentioned and the calculation equations are introduced.

A detailed description of sensitivity calculation method based on ECDs (Element-Coefficient Diagrams) is discussed in Chapter 3. This chapter begins with the brief introduction of DDDs (Determinant Decision Diagrams) and ECDs. Sensitivity equations are obtained by direct differentiation of symbolic expressions stored as ECDs. The proposed methodology has been applied to the calculation of sensitivities of four benchmark circuits. The experiment results and analysis are also demonstrated.

In Chapter 4, small-signal model of the MOSFET for sensitivity analysis is built and the definition of multiparameter is introduced to evaluate the MOSFET sensitivity values. Multiparameter sensitivity algorithm is also implemented. The experimental results are presented for proposed methodology.

Chapter 5 discusses how to apply symbolic performance model and sensitivity weight optimiza-
tion methodology in circuit synthesis. The sensitivity weight optimization algorithm is described and implemented. The experimental results are presented.

Parasitic modeling techniques and parasitic optimization are described in Chapter 6. The behavior of analog circuits is extremely sensitive to the parasitics introduced during the subsequent layout phase. The parasitics can be modeled and optimized to yield fast and accurate parasitic estimates. The optimized parasitic model is used in layout-aware synthesis without performing layout generation which saves significant runtime because of avoiding the time-consuming procedures of layout generation and extraction.

In Chapter 7, we discuss the conclusions and propose directions for future research.
Chapter 2

Sensitivity Analysis Overview

Sensitivities are mathematical measures of the performance metrics due to infinitesimally small perturbations of circuit parameter values [38], which provide additional insight into the behavior of a physical system. There are two main reasons for the study in circuit synthesis:

- Sensitivities help in the understanding of how parasitic parameters of layout or design parameters, such as those of element values, influence the response.

- Sensitivities provide response gradients in optimization applications.

The designer is often interested in the behavior of his circuit when the parameter changes during the synthesis process.

In our approach we consider sensitivities in static circuits which are those described by algebraic equations. This case comprises linear and nonlinear circuits in steady state under DC excitations or small-signal AC excitations. In the DC case, a circuit is described by means of real nonlinear algebraic equations. In the AC case, that circuit is linearized and symbolically transformed into the $j\omega$ domain where it can be described by complex linear equations. Thus, the canonical equations of a static circuit can take real or complex algebraic form. Usually, there are
three main methods to calculate sensitivities: the perturbation method, the direct method and the adjoint method. These three methods can be applied to DC, AC and transient sensitivity computation. Although the details are different, the general principle of the method is the same in the three cases.

2.1 Perturbation Method

The DC solution of a circuit can be computed by solving a system of nonlinear algebraic equations [23]:

\[ f(x^0, p, w) = 0 \]  \hspace{1cm} (2.1)

where \( x^0 \) is the vector of the voltages and currents, \( p \) is the vector of parameters and \( w \) represents DC sources. The purpose of DC sensitivity analysis is to determine the derivative of one or more elements of \( x^0 \) to one or more elements of \( p \).

In the perturbation method one parameter is perturbed by \( \Delta p \) and a new system is formulated and solved to determine \( \Delta x^0 \). The sensitivity is then approximated by the relation:

\[ \frac{\delta x^0}{\delta p} \approx \frac{\Delta x^0}{\Delta p} \]  \hspace{1cm} (2.2)

This brute force approach has several disadvantages. First, the finite difference approximation tends to the differential sensitivity only in the limit as \( \Delta p \to 0 \), and using a very small value for \( \Delta p \) is numerically unstable because of roundoff errors. Second, a complete sensitivity analysis requires the formulation and solution of 2.1 for each parameter of \( p \) which results in prohibitively high computational cost. The advantage of the approach is its straightforward implementation and general applicability [23].
2.2 Direct Method

The simplest direct sensitivity is the derivative of a differentiable function $F$ with respect to any parameter $p$:

$$D^F_p = \frac{\delta F}{\delta p}$$ (2.3)

This definition is useful for computer applications but is not scale free. The most widely used definition is the normalized sensitivity

$$S^F_p = \frac{\delta \ln F}{\delta \ln p} = \frac{p \delta F}{F \delta p} = \frac{p}{F} D^F_p$$ (2.4)

which forms the basis for comparing various designs. When either $p$ or $F$ takes a zero value, the definition 2.4 no longer provides a useful measure, but two other seminormalized sensitivities can be defined [22]:

$$S^F_p = \frac{\delta F}{\delta \ln p} = \frac{p \delta F}{p \delta p} = pD^F_p$$ (2.5)

and

$$S^F_p = \frac{\delta \ln F}{\delta p} = \frac{1 \delta F}{F \delta p} = \frac{1}{F} D^F_p$$ (2.6)

The definition 2.5 is used mainly when $F = 0$, while $S^F_p$ is the natural definition when $p = 0$, that is, when it is a parasitic. When both $F$ and $p$ are zero, 2.3 must be used.

This method generates the sensitivity of the function $F$ with respect to one single variable element $p$. For each additional parameter $p$, the above calculation has to be solved again. In network applications the function $F$ can be a network function or its pole or zero, and so on, while
the parameters $p$ can be component values, the frequency variable $s$, the operating temperature or humidity, and so on.

### 2.3 Adjoint-based Sensitivity Approach

Adjoint-based sensitivity analysis for networks is attributed to Director and Rohrer [9], who derived a sensitivity expression based on Tellegen’s theorem [36]. An adjoint network is constructed through relatively straightforward rules and solved to produce the adjoint voltages and currents, which are subsequently substituted in the sensitivity expression.

The adjoint method involves analyzing two network analyses, one on the original network and the other on a related ‘adjoint network’ to ascertain branch voltages and branch currents. The network sensitivities are found to be simple products of the corresponding branch voltages and/or branch currents so obtained. This method boils down to solving for the rows of the sensitivity matrix instead of the columns as is done in direct approach. Each row corresponds to the sensitivities of a node voltage or branch current with respect to all the design parameters. We define, at a particular frequency, the sensitivity equation can be written as:

$$ JS_{ap} = F $$  \hspace{1cm} (2.7)

where $F$ is a matrix having the same dimensions as that of the sensitivity matrix $S_{ap}$. The rows of $S_{ap}$ matrix are the products of respective rows of $J^{-1}$ and $F$. But the rows of $J^{-1}$ are the columns of $(J^T)^{-1}$ and can be computed by solving the following:

$$ J^T X = E $$  \hspace{1cm} (2.8)
Where $X$ is the desired column vector and $E$ is a column vector with one in the appropriate position (corresponding to the column number associated with $X$) and zero elsewhere. Since the $LU$ factored Jaclbian is available after circuit simulation and $J^T = U^T L^T$, the solution of the above equation simply involves forward and backward substitutions. Details on efficient numerical algorithms for obtaining, storing, and rearranging the $LU$ factors used in sensitivity analysis can be found in [24]. The adjoint method for sensitivity computation is hence more efficient if the number of design parameters exceeds the number of circuit variables whose sensitivities are to be computed. Also this method is very efficient in situations where the sensitivity of one function of the circuit variables to a set of parameters has to be evaluated. However, for time domain sensitivities of the adjoint approach is very cumbersome and involves running the adjoint system backward in time. The direct method on the other hand is much simpler to implement. In this dissertation we use direct method to calculate circuit performance sensitivities with respect to circuit parameters (like capacitances, resistances and MOSFETs) so as to determine the critical parameters.
Chapter 3

Sensitivity Analysis Based on ECDs

3.1 Introduction

Traditionally, analog circuit designers spent a lot of time manually sizing their schematics to meet performance constraints such as gain, unity gain frequency and phase margin. Circuit parameters such as transistor width and capacitance values were determined based on designer’s experience and intuition. The whole tuning and simulation process was tedious, slow and error-prone. Automating the circuit sizing is really crucial to designing high performance and productive circuits.

Sensitivity of an analog circuit is the mathematical measure of variations in the performance metrics due to infinitesimal small perturbations of circuit parameter values. In analog circuit synthesis sensitivity analysis plays an important role in determining the critical design variables. In layout-aware circuit sizing sensitivity analysis can be used repetitively to identify critical parasitics that severely hamper the circuit performance. During circuit optimization sensitivity analysis can be used to obtain response gradients and for comparison between two designs exhibiting similar nominal response. As is evident from these applications, it is imperative to develop a technique that can obtain sensitivity efficiently during each synthesis iteration.
Symbolic sensitivity methods can be used for this purpose by eliminating the use of non-insightful and relatively slow numerical sensitivity analysis tools like SPECTRE and SPICE. Symbolic analysis at the circuit level is a formal technique to calculate the behavior or a characteristic of a circuit with the independent variable. The technique is complementary to numerical analysis (where the variables and the circuit elements are represented by numbers) and qualitative analysis (where only qualitative values are used for voltages and currents, such as increase, decrease or no change) [14]. Symbolic sensitivity analysis is then a formal method to determine the sensitivity of a circuit in terms of symbolic circuit parameters and independent variables like frequency. In the symbolic domain, sensitivity can be represented in an explicit form as the first order derivative of the circuit transfer function with respect to a certain design parameter. These symbolic methods have however suffered from poor scalability and long evaluation time, for even moderately large circuits. The problem is not in the algorithms themselves, but in the underlying symbolic analysis technique used.

With the advent of hierarchical decomposition and the concept of sequence of expressions, significant progress has been made in the symbolic analysis of large circuits [16, 17]. This in turn has led to the development of several symbolic sensitivity analysis techniques based on differentiation of symbolic equations derived by hierarchical symbolic analysis [26, 3, 20, 10]. Each of these methods, however, inherits the limitations of the sequence of expression technique. The expressions generated are not compact enough, do not fully exploit sharing of expressions, cannot be easily manipulated and have larger generation as well as evaluation time compared to the determinant decision diagrams (DDDs) based approach [35].

In this chapter we tackle the aforementioned problems associated with traditional symbolic sensitivity analysis by using the concept of element-coefficient diagrams (ECDs) [13], as the underlying symbolic analysis technique. An ECD is a multi-root version of the DDD (a compact
graphical representation of symbolic determinants [30]) for representing symbolic transfer functions. Each root in an ECD represents a coefficient of the numerator or denominator. The techniques based on DDDs (and hence the ECDs) are the fastest symbolic analysis algorithms reported so far in literature. A DDD-based symbolic sensitivity analysis approach was presented in [30], but it involves dedicated manipulations, like extracting several co-factors from DDDs, and does not exploit the $s$-expanded representation of transfer functions. Moreover, no experiments have been presented to compare the efficiency and accuracy of the algorithm with existing techniques.

In our approach sensitivity expressions are obtained from ECDs by a bottom-up differentiation of each coefficient graph. The differentiation algorithm is applied to each vertex of each coefficient graph in the ECDs. The resulting sensitivity equations are stored as sensitivity-ECDs (SECDs) and inherit all the good properties of ECDs, like sharing of sub-graphs and no term cancellation. The proposed approach has been implemented in C++ and the software tool has been named SSAE (Symbolic Sensitivity Analysis using ECDs). The performance of this tool has been compared to SPECTRE. The comparison has not been made with other existing symbolic sensitivity analysis tools as ECD-based techniques are the most efficient symbolic analysis algorithms.

### 3.2 Overview of Determinant Decision Diagrams (DDDs) and Element-Coefficient Diagrams (ECDs)

Determinant decision diagram is for the generation of exact symbolic network functions in the form of rational polynomials of the complex frequency variable $s$ for analog integrated circuits based on the graph. DDDs represent the sum-of-product expressions. Each vertex in the DDD represents a symbolic determinant and is defined by the determinant expansion and multiplication in normal algebra.
ECD is a fast term generation implementation of the compact representation of determinant decision diagram (DDD [30, 31]). It stores the transfer function as coefficient graphs, allowing overlapping of these graphs (sharing of vertices and edges). As an illustrative example, the DDD graphs of a RC network Figure 3.1 is shown in Figure 3.2, the ECD graphs of the same RC network [13] is shown in Figure 3.3. We can see that each coefficient of the numerator or denominator is represented by one root and has a corresponding coefficient graph. For example the root vertex of the coefficient graph for first coefficient of the numerator is numer_0. The subgraph originating from this vertex is the coefficient graph of the first coefficient of the numerator.

Equation 4.5 shows the transfer function from node \( \text{in} \) to node \( \text{out} \).

\[
H (s) = \frac{V_{\text{out}}}{V_{\text{in}}} = \frac{g_1 g_2}{g_1 g_2 + s (c_2 (g_1 + g_2) + g_2 c_1) + s^2 c_1 c_2} \tag{3.1}
\]

There are two types of special nodes in the ECDs: one is the vertex of out-degree zero such as vertex \( v_1 \) in Figure 3.3; the other is the vertices of in-degree zero which are the vertices of coefficients of the numerator or denominator such as vertex \( \text{numer}_0 \). If the vertex name starts with \( \text{numer} \), it means that it is the coefficient of the numerator. The numbers in these names imply
the degree of the complex frequency \( s \) which the coefficient is associated with. The same rules apply to the denominator.

Every edge in the graph representation is weighted with a single circuit parameter, source or constant. The value of the initial vertex is the sum of products of the weights (circuit parameter) of all the outgoing edges and their corresponding terminal vertices.

Excluding the vertex of out-degree zero, there are three cases of connection between vertices
and we are always only interested in one specific initial vertex.

Case 1: The simplest case, single terminal vertex, single edge shown in Figure 3.4(a).

Given an initial vertex $V_n$ and a terminal vertex $V_{n-1}$ connected by edge with weight $c$ as shown in Figure 3.4.

Let $I_n$ denote the intermediate value of the transfer function at vertex $V_n$. According to the implementation of ECDs, we get:

$$I_n = c \times I_{n-1} \quad (3.2)$$

Note that $I_1=1$.

Case 2: Single terminal vertex, multiple edges from the initial vertex to the terminal vertex shown in Figure 3.4(b).

$$I_n = \sum_{i=1}^{N} c_i \times I_{n-1} \quad (3.3)$$
Case 3: Multiple terminal vertices, single edge from the initial edge to each terminal vertex. Terminal vertices are denoted as \(V_{n-1}^i (i = 1, \cdots, N)\), each edge has a weight of \(c_i (i = 1, \cdots, N)\).

\[
I_n = \sum_{i=1}^{N} c_i \cdot I_{n-1}^i 
\]

(3.4)

In case 3, there can be multiple edges from the initial vertex to any of the terminal vertices shown in Figure 3.4(c). Equation 3.2 will be applied in this case.

The symbolic representation of one coefficient in the numerator or denominator of the transfer function can be easily derived by applying Equation 3.2 ~ 3.4 starting from the node. The following illustrates the steps to calculate \(\text{denom}_\perp I\). Note that the equations are evaluated in the inverse order.

\[
\text{denom}_\perp = -c_2 \cdot I_9 - g_2 \cdot I_{10};
\]

\[
I_{10} = -1 \cdot I_8;
\]

\[
I_9 = -1 \cdot I_7;
\]

\[
I_8 = -c_1 \cdot I_6;
\]

\[
I_7 = (-g_1 - g_2) \cdot I_6;
\]

\[
I_6 = -1 \cdot I_1;
\]

\[
I_1 = 1.
\]

### 3.3 Sensitivity Equations

A symbolic transfer function obtained by symbolic analysis on a linear time-invariant analog circuit is in the form:

\[
H(s, p) = \frac{N(S)}{D(S)} = \frac{\sum_{i=0}^{m} f_i(p) s^i}{\sum_{i=0}^{m} g_i(p) s^i}
\]

(3.5)
where $s$ is the frequency Laplace variable and $p$ denotes the vector of all circuit parameters. The coefficients of the numerator and denominator are represented by $f_i$ and $g_i$ respectively.

The most widely used definition of normalized sensitivity of a transfer function $H$ with respect to a circuit parameter $p_j$ is [38]:

$$S^H_{p_j} = \frac{\partial \ln H}{\partial \ln p_j} = \frac{\partial H}{\partial p_j} \frac{p_j}{H}$$ (3.6)

In order to derive the equation for sensitivity analysis, which is suitable for the use of ECDs, we use the method described in [43]. The following equation is obtained by plugging equation 3.6 in equation 3.5:

$$\frac{\partial H}{\partial p_j} = \frac{\partial \ln H}{\partial \ln D} = \frac{1}{D} \frac{\partial N}{\partial p_j} - \frac{N}{D} \frac{\partial D}{D^2 \partial p_j} = \frac{1}{N} \frac{\partial N}{\partial p_j} - \frac{1}{D} \frac{\partial D}{\partial p_j}$$ (3.7)

Therefore,

$$S^H_{p_j} = \frac{\partial H}{\partial p_j} \frac{p_j}{H} = \frac{p_j}{N} \left( \frac{1}{N} \frac{\partial N}{\partial p_j} - \frac{1}{D} \frac{\partial D}{\partial p_j} \right)$$

$$= \frac{p_j}{N} \sum_{i=0}^{m} s^i \frac{\partial f_i}{\partial p_j} - \frac{p_j}{D} \sum_{i=0}^{n} s^i \frac{\partial g_i}{\partial p_j}$$ (3.8)

From equation 3.8 it is evident that the major task in computing the first order derivative of the transfer function is to compute the first order derivative of each symbolic coefficient of the numerator and denominator. Element-coefficient diagrams store each coefficient of the numerator and denominator as separate coefficient graphs while sharing common sub-graphs. Due to this property of ECDs, it is straightforward to obtain the derivative of each symbolic coefficient.

### 3.4 Sensitivity Analysis Algorithm and Implementation

As shown in Equation 3.8, in order to get the explicit form of sensitivity equation 3.6, it is necessary to calculate the derivative of each coefficient. The following shows how this can be done by simple
Figure 3.5: ECD Representation of Derivative of $I_n$:(Left) When $c = p_j$ (Right) When $c \neq p_j$

graph transformations. These cases correspond to the cases in last section.

Case 1:

The derivative of intermediate value at vertex $V_n$ can be written as:

$$
\frac{\partial I_n}{\partial p_j} = \begin{cases} 
I_{n-1} + c \cdot \frac{\partial I_{n-1}}{\partial p_j}, & \text{if } c = p_j \\
 c \cdot \frac{\partial I_{n-1}}{\partial p_j}, & \text{if } c \neq p_j 
\end{cases}
$$

(3.9)

Thus, the ECD representation of derivative of $I_n$ belongs to one of the two cases shown in Figure 3.5 in which vertex name starting with “D” indicates the derivative of the vertex value.

A special case is:

$$
\frac{\partial I_1}{\partial p_j} = 0
$$

(3.10)

Case 2:

In the single terminal vertex multiple edge case, each edge can be treated individually by splitting the graph in Figure 3.4(b) into $N$ number of graphs in Figure 3.4(a). For each graph the transformation shown in Figure 3.5 is applied. We can combine all $N$ transformed graphs and eliminate the duplicate vertices. This can be done by deleting all the duplicate vertices by one and connect all the edges to or from these vertices to this single vertex.
Case3:

We only consider the case where there is a single edge from the initial vertex to each terminal vertex. The initial vertex is split into $N$ number of graphs as in Case 2. And similarly, we combine the transformed graphs.

The construction of the derivative graph of one coefficient can be done bottom up, applying Equation 3.9 at each vertex until it reaches the top of the graph. The algorithm for construction of the graph is presented below. The function $CreateVertex$ creates a new vertex and links it to already existing vertices. For example $CreateVertex(D_{\downarrow}V_i, D_{\downarrow}V_{i-1}, V_{i-1}, c)$, creates a vertex $D_{\downarrow}V_i$ and links it to vertex $D_{\downarrow}V_{i-1}$ with an edge of weight $c$, and also to vertex $V_{i-1}$ with an edge of weight 1. Similarly $CreateVertex(D_{\downarrow}V_i, D_{\downarrow}V_{i-1}, c)$, creates a vertex $D_{\downarrow}V_i$ and links it to vertex $D_{\downarrow}V_{i-1}$ with an edge of weight $c$. Finally, $CreateVertex(D_{\downarrow}V_i)$ creates a vertex $D_{\downarrow}V_i$ and assigns it a value 0 without linking it to any other vertex. Each $i^{th}$ coefficient graph associated with the numerator is represented by $CGN(i)$ and with denominator by $CGD(i)$. The algorithm presented below is for the construction of SECDs for the ECDs $H$ with respect to parameter $p$. 
Using the algorithm described above, a program called SSAE (Symbolic Sensitive Analysis based on ECDs) has been developed which can perform circuit sensitivity analysis based on the simplified symbolic transfer function represented by ECDs. In SSAE, the derivative graphs are generated and converted to C++ code for faster evaluation.

3.5 Experimental Results

We verify the accuracy and efficiency of our proposed algorithm by comparing our software implementation (SSAE) with SPECTRE. The difference between the magnitude of SPECTRE and
Figure 3.6: Benchmark Circuits (a) Single Transistor Amplifier (b) Single-Ended LNA

Figure 3.7: Two Stage Opamp1
Table 3.1: Maximum Difference in Gain Sensitivity between SSAE and SPECTRE (for RC and STA)

<table>
<thead>
<tr>
<th>Tool</th>
<th>Circuit Name</th>
<th>RC</th>
<th>STA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Sensitivity Real</td>
<td>Sensitivity Imag</td>
</tr>
<tr>
<td>SPECTRE</td>
<td></td>
<td>-8.29047e-12</td>
<td>-1.88496e-07</td>
</tr>
<tr>
<td>SSAE</td>
<td></td>
<td>-8.29047e-12</td>
<td>-1.88496e-07</td>
</tr>
<tr>
<td>Deviation</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.2: Maximum Difference in Gain Sensitivity between SSAE and SPECTRE (for TSO and SE-LNA)

<table>
<thead>
<tr>
<th>Tool</th>
<th>Circuit Name</th>
<th>TSO</th>
<th>SE-LNA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Sensitivity Real</td>
<td>Sensitivity Imag</td>
</tr>
<tr>
<td>SPECTRE</td>
<td></td>
<td>5.77167e+02</td>
<td>1.44174e+03</td>
</tr>
<tr>
<td>SSAE</td>
<td></td>
<td>5.70902e+02</td>
<td>1.44428e+03</td>
</tr>
<tr>
<td>Deviation</td>
<td></td>
<td>2.8576e-3%</td>
<td>3.96861e-6%</td>
</tr>
</tbody>
</table>

SSAE sensitivity results is termed as *deviation*. The sensitivities reported in all tables are not the normalized sensitivities (for compatibility with SPECTRE results) and have a unit *per farad (/F)*.

The comparison is done on four testbenches. The first testbench is a RC network, shown in Figure 3.1. The second testbench is a single transistor amplifier (STA) shown in Figure 6.7(a). The third, SE-LNA shown in Figure 6.7(b), is a single-ended low noise amplifier borrowed from [40]. The fourth is two-stage opamp (TSO) shown in Figure 3.7. For all testbenches only the gain sensitivity has been observed.

In table 3.1 and 3.2 the maximum difference of sensitivity results obtained by SSAE and SPECTRE is reported. For all the testbenches, the sensitivities with respect to all the capacitances are
Table 3.3: TSO Gain Transfer Function Sensitivities w.r.t. C1 for Different Values

<table>
<thead>
<tr>
<th>c1</th>
<th>SSAE Sensitivity Real</th>
<th>SSAE Sensitivity Imag</th>
<th>SPECTRE Sensitivity Real</th>
<th>SPECTRE Sensitivity Imag</th>
<th>deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1e-14</td>
<td>2.181e+2</td>
<td>1.56306e+3</td>
<td>2.24902e+2</td>
<td>1.56205e+3</td>
<td>2.87193e-3%</td>
</tr>
<tr>
<td>1e-13</td>
<td>2.51951e+2</td>
<td>1.55749e+3</td>
<td>2.58726e+2</td>
<td>1.55634e+3</td>
<td>2.43184e-3%</td>
</tr>
<tr>
<td>1e-12</td>
<td>5.70902e+2</td>
<td>1.44428e+3</td>
<td>5.77167e+2</td>
<td>1.44174e+3</td>
<td>2.8576e-3%</td>
</tr>
<tr>
<td>1e-11</td>
<td>6.06e+2</td>
<td>-1.98074e+2</td>
<td>6.05132e+2</td>
<td>-2.00701e+2</td>
<td>4.5276e-3%</td>
</tr>
</tbody>
</table>

Table 3.4: SE-LNA Transfer Function Sensitivity w.r.t. C1 for Different Frequencies

<table>
<thead>
<tr>
<th>Frequency</th>
<th>SSAE Sensitivity Real</th>
<th>SSAE Sensitivity Imag</th>
<th>SPECTRE Sensitivity Real</th>
<th>SPECTRE Sensitivity Imag</th>
<th>deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1e3</td>
<td>2.94576e-10</td>
<td>6.08491e-9</td>
<td>2.94571e-10</td>
<td>6.08491e-9</td>
<td>3.96861e-6%</td>
</tr>
<tr>
<td>1e4</td>
<td>5.67033e-8</td>
<td>6.0848e-6</td>
<td>5.66507e-8</td>
<td>6.08481e-6</td>
<td>1.56278e-4%</td>
</tr>
<tr>
<td>1e5</td>
<td>2.7776e-4</td>
<td>6.0744e-3</td>
<td>2.77239e-4</td>
<td>6.07449e-3</td>
<td>1.08753e-3%</td>
</tr>
<tr>
<td>1e6</td>
<td>2.41792</td>
<td>5.14667</td>
<td>2.41686</td>
<td>5.15239</td>
<td>8.31362e-2%</td>
</tr>
</tbody>
</table>

calculated using both tools. Only the worst case deviation for each testbench is reported. The worst case deviations for RC, STA, TSO and SE-LNA were with respect to C1, C, C1 and C1 respectively. Table 3.3 presents the gain sensitivity for TSO with respect to C1 for different numerical values of C1. Table 3.4 presents the maximum deviation of the SE-LNA gain sensitivity with respect to C1 for frequencies ranging from $10^3$ to $10^6$. The results show that the deviation of sensitivity increases while the frequency grows. This reaches up to 0.08% when the frequency is as high as $10^6$. All the results above show the accuracy of SSAE with respect to SPECTRE as the deviations for all the cases are within 0.1%.

Table 3.5 presents the time results of SPECTRE and SSAE. For SSAE the process of sensitivity analysis involves three parts: generation of sensitivity of ECD (S-ECD) which is a one time process, operating point analysis (using SPECTRE) and evaluation of S-ECD. When SSAE is used
iteratively only the latter two need to be performed and only the last step is termed as an actual sensitivity analysis step. The sensitivity analysis for SPECTRE involved two parts: operating point analysis and the actual process of sensitivity analysis. Both these steps need to be done when SPECTRE is used iteratively. For both tools the gain sensitivity is calculated for 160 frequency points (a reasonable number of frequency points for any AC analysis in the range of 1Hz-10^8Hz) and the difference in time between SPECTRE and SSAE is reported. For sensitivity analysis only SSAE is significantly faster than SPECTRE (21x-35x). Overall SSAE is 19%-30% faster than SPECTRE. To further elucidate the time results Figure 3.8 shows the speedup due to the use of SSAE with respect to SPECTRE for different number of frequency points. The speedup is not significant when sensitivity is calculated for only a few numbers of frequencies, but it increases significantly as the number of frequency points increases. From Figure 3.8 we can observe that sensitivity analysis for larger benchmark circuits such as SE-LNA and TSO shows higher speedup than that for small-scaled circuits like RC and STA. Thus the methodology presented in this chapter is especially efficient for larger circuits compared to SPECTRE. A point to be noted here is that a speedup of only 19%-30% is observed because the operating point analysis takes up a large portion of the time. If this part can be done in SSAE by using faster methods, rather than relying on SPECTRE’s results, a huge speedup will be observed.

Table 3.5: Time Results for Sensitivity Analysis of Gain for 500 Iterations

<table>
<thead>
<tr>
<th>Circuit Name</th>
<th>S-ECD Generation</th>
<th>S-ECD Evaluation</th>
<th>Numerical Analysis</th>
<th>Total</th>
<th>SPECTRE Sensitivity Analysis</th>
<th>Total Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC</td>
<td>0.42s</td>
<td>0.01s</td>
<td>140.77s</td>
<td>141.19s</td>
<td>168.06s</td>
<td>3578.8%</td>
</tr>
<tr>
<td>STA</td>
<td>0.28s</td>
<td>0.01s</td>
<td>141.76s</td>
<td>142.04s</td>
<td>169s</td>
<td>2135.3%</td>
</tr>
<tr>
<td>TSO</td>
<td>0.48s</td>
<td>0.2s</td>
<td>145.57s</td>
<td>146.05s</td>
<td>190.56s</td>
<td>2556.9%</td>
</tr>
<tr>
<td>SE-LNA</td>
<td>0.5s</td>
<td>0.69s</td>
<td>143.25s</td>
<td>143.75s</td>
<td>177.47s</td>
<td>3334.8%</td>
</tr>
</tbody>
</table>
All the above mentioned experiments have been conducted on SunBlade 1000 with Solaris(SunOS), 2048MB RAM and 2-750MHz processors. The experimental results show that SSAE with faster speed has a very small deviation compared to SPECTRE and verify the accuracy and efficiency of the proposed method.

3.6 Conclusions

This chapter presents a sensitivity analysis method via simple differentiation of the ECDs. The algorithm implemented by SSAE can perform circuit sensitivity analysis based on the simplified symbolic transfer function represented by ECDs with limited CPU time and memory space. SSAE outputs the numerical sensitivity values as well as sensitivity equations (fast compilable C++ code) with respect to various circuit parameters. With these outputs, the designer can determine what needs to be done to correct and improve circuit synthesis [8], layout [22] and testing [34] [33]. Since SSAE is based on ECDs, it inherits all the good properties of ECDs such as compact representation and short evaluation time. The use of SSAE speeds up the simulation process considerably especially for larger analog circuits.
Figure 3.8: Speedup of Four Benchmark Circuits under Different Numbers of Frequency in One Iteration
Chapter 4

Multiparameter Sensitivity Modeling and Analysis

4.1 Introduction

As we mentioned in Chapter 3.3 sensitivity of an analog circuit is the mathematical measurement of variations in the performance metrics due to small perturbations of circuit parameter values [22], which plays a very important role in determining the critical design variables. Sensitivity analysis has been used extensively for performance driven layout synthesis in [6] and [5]. The methods which use SPICE iteratively to optimize circuits and calculate sensitivities are computationally expensive and limit the size of the circuit sized in [39, 32]. Sensitivity analysis during a synthesis process is used to calculate the performance sensitivities with respect to capacitors, resistors and the widths of MOSFETs for analog circuits. Compared with the analysis for MOSFETs, sensitivity analysis with respect to capacitors and resistors in an analog circuit is easier. In [7], MOSFET sensitivity was calculated with approximation during circuit sizing and the intrinsic parasitic capacitances of MOSFET at the gate, source and drain were evaluated, but other crucial device model
parameters such as conductances were ignored.

In this chapter we propose a multiparameter sensitivity analysis methodology based on analog small-signal model for MOSFETs [2]. Small-signal model is a linear model that helps in simplifying calculations. We evaluate sensitivities of all the intrinsic parameters including five capacitances and three conductances with a linearized small-signal model for MOSFETs. Multiparameter sensitivity analysis based on small-signal model is more accurate than any other transistor sensitivity analysis methods reported so far in literature.

The techniques to calculate sensitivity value for each intrinsic MOSFET parameter, independent capacitors or resistors are based on determinant decision diagrams. ECDs are the cancellation-free and per-coefficient term generation versions of determinant decision diagrams. ECD based sensitivity analysis called SSAE (symbolic sensitivity analysis based on ECDs) was well developed and implemented in Chapter 3.3. The experimental results are as accurate as numerical sensitivity analysis done using SPECTRE. Moreover, it is more efficient and extremely faster than the latter.

In this chapter we propose multiparameter sensitivity analysis based on element-coefficient diagrams (ECDs) [13] with small-signal model as the underlying symbolic analysis technique. The proposed method is introduced in the analog circuit synthesis framework to reduce the overall number of iterations and runtime. Sensitivity expressions are obtained from ECDs with a bottom-up differentiation of each coefficient graph. The differentiation algorithm is applied to each vertex of each coefficient graph in the ECDs. The resulting sensitivity equations are stored as sensitivity-ECDs (SECDs) which inherit all the good properties of ECDs, like sharing of sub-graphs and no term cancellation. In this chapter we stress how to use efficient SECDs to analyze multiparameter sensitivity based on small-signal MOSFET model. In our approach, multiparameter sensitivity analysis has been implemented and applied to the analog synthesis process. The performance and
efficiency of this algorithm have been demonstrated with the testbench circuit.

The rest of this chapter is organized as follows. The proposed synthesis method is described in Section 4.2. Section 4.3 presents the multiparameter modeling and partial derivative of small-signal model parameter with respect to the width of MOSFETs. Section 4.4 provides a brief overview of multiparameter sensitivity algorithm implementation. Section 4.5 presents the experimental results and Section 4.6 concludes this chapter.

4.2 Proposed Circuit Synthesis Approach

The proposed circuit synthesis environment is shown in Figure 4.1. Circuit topology and search range for each parameter value of the circuit are the inputs to the optimization engine. In this case simulated annealing is used for optimization. A new set of parameter values are passed by the optimization engine to NGSPICE (or other simulators) to perform operating point (OP) analysis and the small-signal values for all active devices generated by OP analysis are passed to the performance evaluator and the multiparameter sensitivity evaluator. The performance evaluator processes the simulation data to obtain the values of the desired performance parameters. These parameters are passed to the cost function evaluator to evaluate the cost which is used by the optimization engine to determine the new device size in each iteration of optimization process. Meanwhile, the multiparameter sensitivity evaluator calculates the performance sensitivity values with respect to all active and passive devices based on small-signal values taken from NGSPICE. It generates a weight vector for each device in the circuit. Instead of randomly choosing the parameter to be perturbed, the odds of devices being chosen for tuning are determined by this weight vector in the optimization loop. This process is repeated until the desired performance goals are achieved.
4.3 Multiparameter Sensitivity Algorithm

In Chapter 3.3, we describe how to analyze performance sensitivity with respect to capacitors and resistors in analog circuits based on ECDs in detail. In this chapter we focus on how to analyze small-signal MOSFET model sensitivity based on ECDs. Compared with other MOSFET sensitivity analysis methods reported so far in literature our approach uses the most accurate MOSFET model and can evaluate the MOSFET sensitivity value precisely. The sensitivities for all the active and passive devices in the circuit can be calculated without approximation. This ensures fast and accurate gradient calculation for circuit synthesis, test and layout.
4.3.1 MOSFET small-signal model multiparameter definition

Generally, sensitivity provides information regarding circuit performance transfer function variation with respect to single-parameter changes. In most cases, circuit transfer function depends on several or all parameters [38].

\[ H = H(p_1, p_2, ..., p_m) = H(p) \] (4.1)

It’s of interest to consider variation in \( H \) when some or all the parameters change simultaneously. The change in \( H \) due to infinitesimally small changes in all parameters is expressed mathematically by the total differential,

\[ dH = \sum_{i=1}^{m} \frac{\partial H}{\partial p_i} dp_i \] (4.2)

In order to introduce normalized sensitivities into equation 4.2, divide by \( H \) and multiply and divide each term inside the summation by \( p_i \):

\[ \frac{dH}{H} = \sum (\frac{\partial H}{\partial p_i} \cdot \frac{p_i}{H}) \cdot \frac{dp_i}{p_i} = \sum S_{p_i}^{H} \cdot \frac{dp_i}{p_i} \] (4.3)

Usually, increments are more instructive:

\[ \frac{\Delta H}{H} \approx \sum S_{p_i}^{H} \frac{\Delta p_i}{p_i} \] (4.4)

By approximating the components in a transistor with similar values of \( \Delta p/p \), we get,

\[ \frac{\Delta H}{H} \approx \frac{\Delta p}{p} \sum S_{p_i}^{H} \] (4.5)
The multiparameter sensitivity (MS) is defined as

\[ MS_k = | \sum S_{p_i}^H | \]  

(4.6)

where the summation is over elements of type \( k \).

For the symbolic MOSFET model we used, there are eight intrinsic parameters which change when the width of a transistor varies. To calculate the performance sensitivity with respect to transistor width, we use the multiparameter sensitivity (MS) method. Based on equation 4.6 deduced above, the computation is performed using the following simplified equation, where \( W_{eff} \) represents MOSFET effective channel width.

\[
\frac{dH}{dW_{eff}} = \frac{\partial H}{\partial g_m} \frac{dg_m}{dW_{eff}} + \frac{\partial H}{\partial g_{mb}} \frac{dg_{mb}}{dW_{eff}} + \frac{\partial H}{\partial g_{rd}} \frac{dg_{rd}}{dW_{eff}} + \frac{\partial H}{\partial C_{bs}} \frac{dC_{bs}}{dW_{eff}} + \frac{\partial H}{\partial C_{gs}} \frac{dC_{gs}}{dW_{eff}} + \frac{\partial H}{\partial C_{gd}} \frac{dC_{gd}}{dW_{eff}} + \frac{\partial H}{\partial C_{bd}} \frac{dC_{bd}}{dW_{eff}}
\]

(4.7)

The right side of the equation above considers all the small-signal model parameters. The derivative of \( H \) with respect to each small-signal parameter such as \( \frac{dg_m}{dW_{eff}} \) can be acquired by SSAE. The key is to find the correlations between effective channel width of MOSFET and all the intrinsic MOSFET small-signal model parameters so that the partial derivative of intrinsic MOSFET parameter with respect to its width such as \( \frac{dC_{gs}}{dW_{eff}} \) can be evaluated. We will show the calculation formulas in the following sections. Section 4.3.2 presents the formulas of eight parameters of small-signal model. Section 4.3.3 presents the partial derivative of the eight small-signal model parameters with respect to MOSFET width.

4.3.2 Modeling of intrinsic MOSFET small-signal model parameters

Small-signal MOSFET model is a linear model and required to obtain network transfer functions that help to simplify calculations, for example the voltage transfer ratio, for a transistor circuit. The
non-linear behavior of a transistor is then linearized at an operating point. The linearization is only valid under the assumption of soft nonlinearities in combination with small signal levels. Symbolic analysis imposes somewhat different requirements on the transistor models than numerical analysis does. A numerical result becomes more detailed with a refined transistor model but the complexity of the result is the same. A symbolic result, on the other hand, becomes much more complicated with the inclusion of parasitic effects. Sometimes we adopt simpler model instead of a detailed one. The difference between a basic MOSFET transistor model and a high-frequency model is shown in Figure 4.2 and Figure 4.3. The basic model considers the transconductance, the output conductance and the input capacitance. The high-frequency model is charge-oriented and includes transcapacitances to model the nonreciprocal capacitance behavior. Also included are a number of parasitic capacitances. It’s possible to find medium-frequency models that are a compromise between these two extremes [11].

![Figure 4.2: Basic MOS Transistor Model](image)

Figure 4.4 shows MOSFET LEVEL 3 model of SPICE which includes a number of parasitics. This model is most popular model used in circuit synthesis.

The conductances \( g_{bd} \) and \( g_{bs} \) are the equivalent conductances of the bulk-drain and bulk-to-source junctions. Since these junctions are normally reverse biased, the conductances are very
small. They are defined as:

\[ g_{bd} = \frac{\partial i_{BD}}{\partial v_{BD}} \text{(evaluated at the quiescent point)} \approx 0 \]  \hspace{1cm} (4.8)

and

\[ g_{bs} = \frac{\partial i_{BS}}{\partial v_{BS}} \text{(evaluated at the quiescent point)} \approx 0 \]  \hspace{1cm} (4.9)

The channel conductances \( g_m \), \( g_{mb} \), and \( g_{ds} \) are defined as:

\[ g_m = \frac{\partial i_D}{\partial v_{GS}} \text{(evaluated at the quiescent point)} \approx 0 \]  \hspace{1cm} (4.10)
Furthermore we will deduce all the equations of conductances and capacitances used in the small-signal model.

In SPICE LEVEL 3 model for MOSFET, which we have used in this work, there is a general

\[ g_{mb} = \frac{\partial i_D}{\partial V_{BS}} \text{(evaluated at the quiescent point)} \approx 0 \]  \quad (4.11)

\[ g_{ds} = \frac{\partial i_D}{\partial V_{DS}} \text{(evaluated at the quiescent point)} \approx 0 \]  \quad (4.12)
equation for $I_{DS}$ when $V_{GS} > V_T$ [2]

$$I_{DS} = \beta(V_{GS} - V_{TH} - \frac{1 + F_B}{2} V_{DS}) V_{DS}$$  \hspace{1cm} (4.13)

Where,

$$F_B = \frac{\gamma F_S}{4\sqrt{PHI - V_{BS}}} + F_N$$

$$\beta = \mu_{eff} C_{ox} \frac{W_{eff}}{L_{eff}}$$

So the channel conductances $g_m$, $g_{mb}$, and $g_{rd}$ can be deduced by their definitions:

$$g_m = \frac{dI_{DS}}{dV_{GS}} = \beta V_{DS}$$  \hspace{1cm} (4.14)

$$g_{mb} = \frac{dI_{DS}}{dV_{BS}} = \frac{dI_{DS}}{dF_B} \frac{dF_B}{dV_{BS}}$$

$$= - \frac{\beta V_{DS}^2}{2 \cdot 4\sqrt{(PHI - V_{BS})^3}} \frac{\gamma F_S}{1}$$

$$= - \frac{\beta \gamma V_{DS}^2 F_S}{16\sqrt{(PHI - V_{BS})^3}}$$  \hspace{1cm} (4.15)

$$g_{rd} = \frac{dI_{DS}}{dV_{DS}}$$

$$= \beta [(V_{GS} - V_{TH}) - (1 + F_B)V_{DS}]$$  \hspace{1cm} (4.16)

To determine the capacitance values for small-signal model, the capacitances can be classified into two categories. The first type includes capacitances $C_{gb}$, $C_{gs}$ and $C_{gd}$ which are all common to the gate and are dependent on the operating condition of the transistors. The second type includes capacitances $C_{bd}$ and $C_{bs}$, which are associated with the back-biased depletion region between the drain and substrate and the source and substrate.

We shall use the following formulas for the first type capacitances of the MOS device in the indicated regions.

Off
\[
C_{gb} = C_{ox}(W_{eff})(L_{eff}) + CGBO(L_{eff}) \quad (4.17)
\]
\[
C_{gs} = CGSO(W_{eff}) \quad (4.18)
\]
\[
C_{gd} = CGDO(W_{eff}) \quad (4.19)
\]

**Saturation**

\[
C_{gb} = CGBO(L_{eff}) \quad (4.20)
\]
\[
C_{gs} = \frac{2}{3} C_{ox}(W_{eff})(L_{eff}) + CGSO(W_{eff}) \quad (4.21)
\]
\[
C_{gd} = CGDO(W_{eff}) \quad (4.22)
\]

**Nonsaturated**

\[
C_{gb} = CGBO(L_{eff}) \quad (4.23)
\]
\[
C_{gs} = (0.5 C_{ox} L_{eff} + CGSO)(W_{eff}) \quad (4.24)
\]
\[
C_{gd} = (0.5 C_{ox} L_{eff} + CGDO)(W_{eff}) \quad (4.25)
\]

Where CGBO, CGSO and CGDO are constants for indicated MOSFET models.

For the second type of capacitances, we consider both the bottom and sidewall components.

The formulas are given as follows [2]:

\[
C_{bx} = \frac{(CJ)(AX)}{[1 - \left( \frac{v_{bx}}{P_B} \right)]^{MJ}} + \frac{(CJSW)(PX)}{[1 - \left( \frac{v_{bx}}{P_B} \right)]^{MJSW}}, \quad (4.26)
\]
\[
v_{BX} \leq (FC)(PB)
\]
\[
C_{bx} = \frac{(CJ)(AX)}{(1 - FC)^{1+MJ}[1 - (1 + MJ)FC + MJ \frac{v_{BX}}{PB}]} + \frac{(CJSW)(PX)}{(1 - FC)^{1+MJSW}[1 - (1 + MJSW)FC + MJSW \frac{v_{BX}}{PB}]}, \quad (4.27)
\]
\[
v_{BX} \geq (FC)(PB)
\]

Where \( x = d \) for \( C_{bd} \) or \( x = s \) for \( C_{bs} \).
4.3.3  Partial derivative of small-signal model parameters of MOSFET with respect to its width

It is not hard to deduce the partial derivative equations of MOSFET intrinsic parameters with respect to its width. The resulting equations in the indicated regions are shown below:

\[
\frac{dg_m}{dW} = V_{DS} \frac{d\beta}{dW} = \frac{V_{DS} \mu_{eff} C_{ox}}{L_{eff}} \tag{4.28}
\]

\[
\frac{dgmbs}{dW} = - \frac{\frac{d\beta}{dW}}{dW_{eff} \frac{16}{(PHI - V_{BS})^3}} \gamma V_{DS}^2 F_S \tag{4.29}
\]

\[
\frac{dgdss}{dW} = \frac{\frac{d\beta}{dW}}{L_{eff} \frac{16}{(PHI - V_{BS})^3}} [V_{GS} - V_{TH}] - (1 + F_B) V_{DS} \tag{4.30}
\]

**Off**

\[
\frac{dC_{gb}}{dW} = C_{ox} L_{eff} \tag{4.31}
\]

\[
\frac{dC_{gs}}{dW} = CGSO \tag{4.32}
\]

\[
\frac{dC_{gd}}{dW} = CGDO \tag{4.33}
\]

**Saturation**

\[
\frac{dC_{gb}}{dW} = 0 \tag{4.34}
\]

\[
\frac{dC_{gs}}{dW} = \frac{2}{3} C_{ox} L_{eff} + CGSO \tag{4.35}
\]

\[
\frac{dC_{gd}}{dW} = CGDO \tag{4.36}
\]
For any indicated region,

\[
\frac{dC_{bs}}{dW} = 0 \quad (4.40)
\]
\[
\frac{dC_{bd}}{dW} = 0 \quad (4.41)
\]

After all the above derivative equations of the eight intrinsic MOSFET small-signal model parameters in the indicated regions were deduced, we introduced them in equation 4.7 to evaluate MOSFET sensitivities accurately.

### 4.4 Multiparameter Sensitivity Analysis Implementation in Analog Circuit Synthesis

Multiparameter sensitivity analysis of performance parameter with respect to the circuit parameters helps in determining which parameters are crucial towards performance degradation and which ones are relatively insignificant. We adopt the proposed multiparameter sensitivity analysis algorithm to calculate the performance sensitivity with respect to every parameter in the circuit. The obtained gradient information is used to determine the crucial parameters to be tuned during the device sizing process.

The core of multiparameter sensitivity analysis is a library of pre-complied symbolic sensitivity equations generated by SSAE. Symbolic analysis has been widely applied for analog circuit
analysis and synthesis. It is a formal technique used to obtain network transfer functions in terms of symbolic circuit parameters and independent variables like frequency [12]. The performance transfer functions of a circuit are stored in the s-polynomial format as ECDs. We use ECDs to evaluate circuit performance sensitivity with respect to each parameter. The process of ECD-based symbolic sensitivity analysis is described in Chapter 3.3. The generated sensitivity equations are converted to C++ code which is then compiled. The use of pre-compiled sensitivity equations based on ECDs reduces the multiparameter sensitivity evaluation time considerably.

After the multiparameter sensitivity analysis, a parameter weight vector is generated to determine the odds of a parameter being chosen for tuning by the optimization engine. The parameter weight vector is based on parameter sensitivity values. The weight of each parameter is proportional to its sensitivity value. For example, n variables in a circuit are described as \( V \{ V_1, V_2, ..., V_n \} \). Multiparameter sensitivity evaluator generates a weight vector for all variables according to their sensitivity values, which is described as \( W \{ W_{V_1}, W_{V_2}, ..., W_{V_n} \} \). \( W_{V_i} \) is proportional to the sensitivity value of \( V_i \). The weight vector is passed to optimization engine to determine the possibilities of each variable to be chosen for tuning in the next optimization loop. The loop repeats until the performance constraint specifications meet.

The experimental environment was set up and the proposed approach was evaluated. An average speedup was observed as discussed in the next section for two stage opamp.

### 4.5 Experimental Results

We verify the efficiency of our proposed algorithm by implementing it in the analog synthesis process. The comparison is done on a two stage opamp circuit shown in Figure 3.7.

In Table 4.1, we present two sets of synthesis results, one is without proposed method (WO in Table 4.1), and the other is with the proposed method (WS in Table 4.1).
Table 4.1: Synthesis Result Comparison without/with Multiparameter Sensitivity Analysis

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Performance Specification</th>
<th>seed</th>
<th># of iterations</th>
<th>Iteration</th>
<th>Runtime(s)</th>
<th>Speedup(%)</th>
<th>Runtime</th>
<th>Speedup(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>WO</td>
<td>WS</td>
<td>WO</td>
<td>WS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSO</td>
<td>Gain(dB) weight(60) ≥ 40</td>
<td>seed1</td>
<td>144</td>
<td>74</td>
<td>48.6</td>
<td>124.9</td>
<td>80.1</td>
<td>35.9</td>
</tr>
<tr>
<td></td>
<td>Phase Margin(°) weight(20) ≥ 60</td>
<td>seed2</td>
<td>290</td>
<td>244</td>
<td>15.8</td>
<td>369.7</td>
<td>322</td>
<td>12.9</td>
</tr>
<tr>
<td></td>
<td>3dB(MHz) weight(15) ≥ 5</td>
<td>seed3</td>
<td>174</td>
<td>99</td>
<td>43.1</td>
<td>151.2</td>
<td>111.3</td>
<td>39.9</td>
</tr>
<tr>
<td></td>
<td>UGF(MHz) weight(5) ≥ 50</td>
<td>seed4</td>
<td>344</td>
<td>213</td>
<td>38.08</td>
<td>489.9</td>
<td>362.1</td>
<td>26.1</td>
</tr>
<tr>
<td>TSO</td>
<td>Gain(dB) weight(60) ≥ 30</td>
<td>seed1</td>
<td>135</td>
<td>89</td>
<td>34.07</td>
<td>113.5</td>
<td>89.1</td>
<td>21.5</td>
</tr>
<tr>
<td></td>
<td>Phase Margin(°) weight(20) ≥ 60</td>
<td>seed2</td>
<td>220</td>
<td>190</td>
<td>13.6</td>
<td>269.7</td>
<td>222.1</td>
<td>17.7</td>
</tr>
<tr>
<td></td>
<td>3dB(MHz) weight(15) ≥ 2</td>
<td>seed3</td>
<td>134</td>
<td>82</td>
<td>38.8</td>
<td>101.9</td>
<td>70.1</td>
<td>31.2</td>
</tr>
<tr>
<td></td>
<td>UGF(MHz) weight(5) ≥ 50</td>
<td>seed4</td>
<td>344</td>
<td>204</td>
<td>46.1</td>
<td>323.9</td>
<td>200.2</td>
<td>38.2</td>
</tr>
</tbody>
</table>

The experiments are conducted with two sets of performance specifications, each set of performance specifications includes gain, phase margin, 3db and unity gain frequency (UGF) as shown in Table 4.1. We can assign different weight for each performance specification according to the design requirements. In our experiments, the weights for gain, phase margin, 3db and UGF are 60%, 20%, 15% and 5% separately. The sum of these weights adds up to 100%. Seed is used to generate random numbers in optimization engine process. For every execution a different seed value was chosen to allow for an unbiased comparison.

In the above experiments we evaluated the number of iterations and runtime for each synthesis process. Considering eight situations with various performance specifications and seed selections, the average of iteration and runtime speedup by using the proposed algorithm is larger than 30%. The proposed approach significantly reduces the synthesis time and the number of iterations.

All the experiments mentioned above have been conducted on SunBlade 1000 with Solaris (SunOS), 2048MB RAM and 2-750MHz processors. The experimental results show a considerable
speedup in analog synthesis with multiparameter sensitivity analysis.

### 4.6 Conclusions

This chapter presents a multiparameter sensitivity analysis method via simple differentiation of the ECDs and its application in analog circuit synthesis. The proposed algorithm performs sensitivity analysis to tune circuit parameters during sizing based on the simplified symbolic transfer function represented by ECDs. The algorithm analyzes performance sensitivity with respect to various circuit parameters. Especially the combination of the intrinsic parameters of MOSFETs which is the most accurate MOSFET sensitivity evaluation reported so far. With this analysis the parameter weight vector generated by proposed approach can control the odds of which parameter may be chosen for tuning during synthesis instead of choosing it randomly. The obtained gradient information can be used to determine what needs to be done to correct and improve in circuit synthesis, testing and layout. Since the proposed algorithm is based on ECDs, it inherits all the good properties of ECDs such as compact representation and short evaluation time. The use of multiparameter sensitivity analysis speeds up the sensitivity computation and sizing process considerably for analog circuits.
Chapter 5

Sensitivity Weight Optimization Algorithm in Analog Circuit Synthesis

5.1 Introduction

Analog circuit synthesis is the process of determining numerical values for the unsized circuit devices of a fixed circuit topology while satisfying a set of performance constraints [28]. Traditionally, analog circuit designers spent a lot of time manually sizing their schematics to meet performance constraints such as gain, unity gain frequency and phase margin. Designers determined and sized the circuit parameters such as transistor width and capacitance values just by their experience and intuition. The whole tuning and simulation process were tedious, slow and error-prone. Automating the circuit optimization process is an important step towards rapidly designing high performance, custom circuits.

In this chapter we propose sensitivity weight optimization algorithm and symbolic performance evaluation in synthesis loop. The proposed method is introduced in the analog circuit synthesis framework to reduce the overall number of iterations and runtime. Both sensitivity calculation and
circuit performance evaluation expressions are obtained from ECDs with a bottom-up differentiation of each coefficient graph which are described in Chapter 3.3. We apply those pre-compiled expressions in a circuit synthesis loop instead of introducing simulation tools like SPICE, NGSPICE or SPECTRE again and again to do calculations numerically.

In this chapter we stress how to use efficient sensitivity weight optimization algorithm and circuit performance evaluation method to speed up the circuit synthesis process. In the circuit synthesis process we adopt different sensitivity weight according to the different synthesis phases (temperatures based on the simulated annealing algorithm). In our approach, proposed methodology has been implemented and applied to the analog synthesis process. The performance and efficiency of this algorithm have been demonstrated with the benchmark circuits.

The rest of this chapter is organized as follows. The proposed analog circuit synthesis approach is described in Section 5.2, which presents sensitivity calculation methods for devices in the circuit and sensitivity weight optimization algorithm. Section 5.3 provides a brief overview of sensitivity weight optimization algorithm implementation. Experimental results are presented in Section 5.4 followed by conclusions in Section 5.5.

5.2 Proposed Circuit Synthesis Approach

The proposed circuit synthesis environment is shown in Figure 5.1. Circuit topology and search range for each parameter value of the circuit are the inputs to the optimization engine. In this case simulated annealing is core algorithm used for optimization which generates a new set of parameter values. Optimization engine then pass those new set of parameter values to NGSPICE (or other simulator) to perform operating point (OP) analysis and the small-signal values for all active devices generated by OP analysis are passed to the symbolic performance evaluator and the symbolic sensitivity evaluator. The symbolic performance evaluator processes the simulation data
to obtain the values of the desired performance parameters. These parameters are passed to the
cost function evaluator to evaluate the cost which is used by the optimization engine to judge if the
specifications meet in each iteration process of optimization. Meanwhile, the symbolic sensitivity
evaluator updates the performance sensitivity values by pre-compiled expresses with respect to
all active and passive devices based on small-signal values taken from NGSPICE. It generates
a sensitivity weight vector including each device in the circuit. The sensitivity weight vector is
passed to optimization engine to update new set of parameter values. Instead of randomly choosing
the parameter to be perturbed, the odds of devices being chosen for tuning are determined by this
sensitivity weight vector in the optimization loop. In the different synthesis phases the sensitivity
weight vectors applied to determine the device to be tuned vary by multiplying a sensitivity rate.
This process is repeated until the desired performance goals are achieved.

5.2.1 Symbolic sensitivity calculation based on ECDs

In a synthesis loop, the sensitivity values of all devices in a circuit should be carefully evaluated
which include passive and active devices. Sensitivity analysis of passive devices such as capaci-
tors and resistors is presented in [42], where there are detailed descriptions about how to analyze
performance sensitivity with respect to capacitors and resistors in analog circuits based on ECDs.
An ECD is a multi-root version of the DDD (a compact graphical representation of symbolic de-
terminants [30]) for representing symbolic transfer functions. Each root in an ECD represents a
coefficient of the numerator or denominator. Sensitivity expressions are obtained from ECDs by
a bottom-up differentiation of each coefficient graph. The differentiation algorithm is applied to
each vertex of each coefficient graph in the ECDs. The resulting sensitivity equations are stored
as sensitivity-ECDs (SECDs) and inherit all the good properties of ECDs, like sharing of sub-
graphs and no term cancellation. SECDs can be converted to C++ code for faster evaluation in the
Compared with passive devices, the sensitivity analysis of MOSFET is more complicated. There are MOSFET sensitivity analysis methods reported in literature, but most of these analyses are not accurate. In [7], MOSFET sensitivity was calculated with approximation during circuit sizing and the intrinsic parasitic capacitances of MOSFET at the gate, source and drain modes were evaluated, but other crucial device model parameters such as conductances were ignored. In this chapter, we adopt multiparameter sensitivity (MS) method described in [41], since MS uses the accurate symbolic MOSFET model and also can evaluate the MOSFET sensitivity value very fast. The precise sensitivity evaluations for all the active and passive devices in the circuit without
approximation ensure fast and accurate gradient calculation for circuit synthesis, test and layout. MS is based on small-signal MOSFET model (SPICE LEVEL 3). For the symbolic MOSFET model used, there are eight intrinsic parameters which change when the width of a transistor varies. That means we should consider all intrinsic parameters when we evaluate the sensitivity of the transistor. The core of MS is equation described in Chapter 4.3 which is shown as follows, where $W_{eff}$ represents MOSFET effective channel width [41].

$$\frac{dH}{dW_{eff}} = \frac{\partial H}{\partial g_m} \frac{d g_m}{d W_{eff}} + \frac{\partial H}{\partial g_{mb}} \frac{d g_{mb}}{d W_{eff}} + \frac{\partial H}{\partial g_{rd}} \frac{d g_{rd}}{d W_{eff}} + \frac{\partial H}{\partial C_{id}} \frac{d C_{id}}{d W_{eff}}$$

$$+ \frac{\partial H}{\partial C_{bs}} \frac{d C_{bs}}{d W_{eff}} + \frac{\partial H}{\partial C_{gs}} \frac{d C_{gs}}{d W_{eff}} + \frac{\partial H}{\partial C_{gd}} \frac{d C_{gd}}{d W_{eff}} + \frac{\partial H}{\partial C_{gb}} \frac{d C_{gb}}{d W_{eff}}$$

Sensitivity calculation expressions for all passive and active devices based on the sensitivity analysis above are stored as SECDs which can be converted into C++ code and then compiled as shown in Figure 5.2. The approach that uses these symbolic expressions in synthesis loop instead of introducing SPICE iteratively to calculate sensitivities speeds up the synthesis process significantly.

### 5.2.2 Symbolic performance evaluator

A major obstacle of automatic sizing, in practice, is that the simulations based on numerical method in traditional circuit synthesis methods are computationally intensive and thus time consuming. It can be avoided by using symbolic performance evaluation. In our work symbolic analysis is used for the generation of symbolic performance models. Symbolic performance models are a combination of transfer functions and that yield a performance parameter when the values of small signal parameters and frequency are fed into it.

Symbolic analysis calculates the behavior or characteristic of a circuit in terms of symbolic parameters. In contrast to numerical simulators such as SPICE that only provide numerical results,
symbolic simulators can explicitly express which circuit parameters determine the circuit behavior. They can offer more advantages than numerical simulators in many applications such as optimum topology selection, behavioral model generation and circuit performance modeling.

Symbolic circuit functions maybe advantageous for repetitive numerical evaluation. For example, in the extreme case that $s$ is the only symbolic variable, evaluation of a $s$ polynomial for frequency-domain simulation can be much faster than solving repeatedly a set of circuit equations.

Symbolic performance models act two roles in our proposed synthesis methodology. One is called symbolic performance evaluator which is used to estimate analog circuit performance and represented by symbolic expressions obtained from symbolic transfer functions (represented by ECDs). In analog circuit synthesis, symbolic performance evaluator is used for repetitive per-
formance estimation during the optimization iterations. The other is called symbolic sensitivity evaluator used to evaluate performance sensitivities with respect to circuit parameters. The two evaluators are shown in Figure 5.1.

The framework of the symbolic performance evaluator is presented in Figure 5.2. The first step is the symbolic analysis engine uses circuit topology and search range for each parameter value of the circuit to generate required symbolic transfer function as Element-Coefficient Diagrams. The second step is symbolic sensitivity and performance analysis engine generates the formulae for the desired performance characteristics (such as gain in this chapter). The obtained formulae are represented in C++ code then compiled for later use.

The core of symbolic performance evaluator is symbolic expressions which obtained from ECDs and converted to C++ code and then compiled. The use of pre-compiled ECDs reduces the symbolic performance evaluation time considerably. The most important thing is that symbolic performance and sensitivity calculation expressions are obtained at the same time, which is crucial to reducing runtime during the synthesis process.

5.2.3 Sensitivity weight optimization algorithm

Simulated annealing based optimization engine determines device sizes in each iteration process of the optimization loop. The procedure Simulated annealing simply invokes MoveFunc at various (decreasing) temperatures. The amount of time spent in annealing at a temperature is gradually increased as temperature is lowered. This is done using the parameter $\beta > 1$. The variable Time keeps track of the time being expended in each call to the MoveFunc. The annealing procedure halts when Time exceeds the allowed time or Cost equals to zero.

The core of sensitivity weight optimization methodology is to assign various sensitivity weight vectors in different annealing procedures based on the temperature. In high temperature phase, we
assign large sensitivity weights on the devices with high sensitivity values and small weights on those devices with low sensitivity values. In other words, the device with high sensitivity value has high possibility to be chosen for tuning in the next sizing iteration. In the moderate temperature phase, we lower the sensitivity weights of the devices with high sensitivity values and increase the weights of the devices with low sensitivity values. In the low temperature phase we disable weight vector based on sensitivity analysis and choose the device to be tuned randomly. We describe the functions used in the algorithm in details as in the following:

**Cost Function** In order to use simulated annealing to solve the problem, we will formulate a cost function which reflects if the desired performance goals are achieved. In this chapter, we define:

\[
Cost(S) = (Gain(\text{New}S) - Gain(S)) \times 100
\]  

(5.1)

In Equation 5.1 the value of \(Gain()\) is generated by symbolic performance evaluator which is a symbolic cost evaluation method instead of introducing simulators like SPICE iteratively to evaluate the performance. \(Cost\) is evaluated by the cost function evaluator shown in Figure 5.1. When \(Gain\) meets the performance goal, \(Cost\) equals to zero.

**Move Function** The core of simulated annealing is \(MoveFunc\) in our approach, which simulates the circuit device sizing process at a given (decreasing) temperature \(T\). The temperature is initialized to a value \(T_0\) at the beginning of the procedure and is slowly reduced during the synthesis loop.

**SenS Function** \(SenS\) function is used by simulated annealing to generate a new set of parameter sizes. Once \(MoveFunc\) is invoked, a device is chosen and assigned a new value. Parameter sizes are then updated for performance evaluation by the symbolic performance evaluator. The device
chosen for sizing is not picked randomly. The odds of devices being chosen for tuning are determined by the sensitivity weight vector $SensVec$. Since the sensitivity weight of each parameter is proportional to its sensitivity value, the device with large sensitivity value has high possibility to be chosen for tuning.

Below is the detailed algorithm of simulated annealing based on sensitivity weight optimization methodology.
Algorithm Simulated\_annealing\((S_0, T_0, T_m, T_l, \alpha, \beta, w, Maxtime, SensVector)\)

\((S_0\text{ is the initial solution})\)

\((T_0\text{ is the initial temperature})\)

\((T_m\text{ is the medium temperature})\)

\((T_l\text{ is the low temperature})\)

\((\alpha\text{ is the cooling rate})\)

\((\beta\text{ is a constant})\)

\((Maxtime\text{ is the total allowed time for the annealing process})\)

\((SensVector\text{ is sensitivity weight vector used to determine the possibility of each device chosen for tuning in MoveFunc})\)

\((w\text{ is SensVector rate})\)
Begin

$T = T_0$;

$S = S_0$;

$Time = 0$;

Repeat

If $(T_0 \geq T \geq T_m)$

Then $MoveFunc(S, T, w * w * SensVector)$;

If $(T_m > T \geq T_i)$

Then $MoveFunc(S, T, w * SensVector)$;

If $(T < T_i)$

Then $MoveFunc(S, T, SensVector)$;

Until $((Time \geq MaxTime) \lor (Cost = 0))$
5.3 Analog Circuit Synthesis Implementation with Proposed Approach

Optimization engine passes parameter values to NGSPICE to perform OP analysis and the small-signal values for all active devices are generated. The symbolic performance evaluator processes the small-signal values to obtain the values of the desired performance parameters and the cost evaluator evaluates the performance parameter to generate the cost which is used to determine new set of parameter sizes. At the same time, symbolic sensitivity evaluator generates a set of sensitivity values for each device needed to be sized and then the sensitivity weight vector is
generated according to the device’s sensitivity values in the circuit, which determines the odds of a parameter being chosen for tuning by the optimization engine. Instead of randomly choosing the parameter to be perturbed the odds of devices being chosen for tuning are determined by this sensitivity weight vector in the optimization loop.

The sensitivity weight of each parameter is proportional to its sensitivity value. For example, n variables in a circuit are described as \( V \{ V_1, V_2, ..., V_n \} \), symbolic sensitivity evaluator generates a sensitivity weight vector for all variables according to their sensitivity values, which is described as \( W \{ W_1 * V_1, W_2 * V_2, ..., W_n * V_n \} \). \( W_i \) is proportional to the sensitivity value of \( V_i \). For the same set of sensitivity values, the sensitivity weight vector \( W_i \) can vary according to design requirement. It means that for the same set of sensitivity values there are several sets of weight vectors accordingly.

Finally, the weight vectors generated are passed to optimization engine to determine the possibilities of each variable to be chosen for tuning in the next optimization loop. The loop repeats until the performance constraint specifications meet.

### 5.4 Experiments

Synthesis experiments were performed on our three benchmark circuits. The first benchmark is two stage opamp shown in Figure 3.7. The second benchmark is yet another two stage opamp shown in Figure 5.3(a). Single-ended opamp(SEO) is the device model of CMOS operational amplifier shown in Figure 5.3(b). For all performance specifications only gain has been observed for simplified purpose.

Table 5.1 gives an account of the number of transistors, capacitors and resistors included in each benchmark circuit. Table 5.2 presents the speedup of iteration and runtime results between traditional method and proposed methodology. In Table 5.2, WS stands for the traditional synthesis process without sensitivity analysis and PS is the proposed synthesis with sensitivity weight op-
Figure 5.3: Two Benchmark Circuits Used in the Experiments

Optimization algorithm. Sensitivity analysis in circuit synthesis reduces the iterations and hence the runtime. From Table 5.2 we can see that the average iteration speedup for three benchmark circuits is around 27% because of the application of sensitivity weight optimization algorithm analysis. Symbolic performance evaluation only can reduce the total runtime, which can not increase or decrease the total iterations. The results show that the average runtime is 59.5% which is reduced greatly by using symbolic performance evaluation and sensitivity weight optimization algorithm.

In our proposed methodology, sensitivity weight varies during the different synthesis phase based on the temperature. In order to verify the efficiency of our proposed method we assign same sensitivity weight instead of variable sensitivity weight in the whole synthesis process for comparison. Table 5.3 presents the time results for proposed comparison experiment. In Table 5.3, SW means the synthesis method using the same sensitivity weight. From Table 5.3 we can see that the average speedup of runtime is 52.3% and the average iteration speedup is 14.5%. Table 5.3 shows sensitivity analysis with the same weight during the synthesis process can improve the synthesis efficiency, but it is not as efficient as sensitivity weight optimization algorithm.
<table>
<thead>
<tr>
<th>Circuit Name</th>
<th>Transistors(#)</th>
<th>Capacitors(#)</th>
<th>Resistors(#)</th>
<th>ECD-Vertices(#)</th>
<th>ECD-Edges(#)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSO1</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>352</td>
<td>1193</td>
</tr>
<tr>
<td>TSO2</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>770</td>
<td>4453</td>
</tr>
<tr>
<td>SEO</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>406</td>
<td>1982</td>
</tr>
</tbody>
</table>

Table 5.2: Iteration and Runtime Comparison without/with Sensitivity Weight Optimization Algorithm

<table>
<thead>
<tr>
<th>Circuit Name</th>
<th>Performance Specification</th>
<th>Number of Iterations</th>
<th>Iteration Speedup(%)</th>
<th>Runtime(s)</th>
<th>Runtime Speedup(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>WS</td>
<td>PS</td>
<td></td>
<td>WS</td>
</tr>
<tr>
<td>TSO1</td>
<td>gain ≥ 40</td>
<td>714</td>
<td>510</td>
<td>28.57</td>
<td>306</td>
</tr>
<tr>
<td>TSO2</td>
<td>gain ≥ 50</td>
<td>1481</td>
<td>1063</td>
<td>28.2</td>
<td>636.2</td>
</tr>
<tr>
<td>SEO</td>
<td>gain ≥ 45</td>
<td>624</td>
<td>471</td>
<td>24.5</td>
<td>270.1</td>
</tr>
</tbody>
</table>

Overall, the performance convergence in both cases is fast and sensitivity weight optimization algorithm and symbolic performance evaluation can improve the synthesis efficiency considerably (around 60%).

All the experiments mentioned above have been conducted on SunBlade 1000 with Solaris (SunOS), 2048MB RAM and 2-750MHz processors. The experimental results show a considerable speedup in analog synthesis with multiparameter sensitivity analysis.

### 5.5 Conclusions

The chapter has proposed a new methodology for sensitivity weight optimization algorithm analysis in analog synthesis. The use of symbolic performance evaluation and sensitivity weight optimization algorithm speeds up the synthesis process significantly and the overall synthesis speed
is faster by 58%-60%. Pre-compiled symbolic performance evaluation and sensitivity calculation expressions are generated at the same time which reduces runtime greatly. The evaluation of pre-compiled symbolic performance and sensitivity analysis models is considerably faster than those in memory.
Chapter 6

Efficient Symbolic Sensitivity based
Parasitic-Inclusive Optimization in
Layout-Aware Analog Circuit Synthesis

6.1 Introduction

Analog circuit synthesis is the process of determining numerical values for the unsized circuit devices of a fixed circuit topology while satisfying a set of performance constraints [28]. The behavior of analog circuits is extremely sensitive to the parasitic effects introduced during the subsequent layout phase. Parasitics not only influence the circuit performance but also lead to the failure of the circuit if it is optimized without taking layout effects into account.

The problem of performance degradation due to layout parasitics can be alleviated by modeling the layout effects during the circuit design or synthesis. This solution increases the total synthesis time due to the large number of parasitics included in the synthesis loop. When parasitic effects are considered, the complete circuit usually becomes much too complicated for hand analysis so
finding the optimum solution is nearly impossible. Designers frequently have to decide among a large number of possible solutions representing different trade-offs. This motivates the need for parasitic aware optimization.

Sensitivity of an analog circuit is the mathematical measurement of variations in the performance metrics due to small perturbations of circuit parameter values as well as parasitic effects, which plays a very important role in determining the critical design variables. Sensitivity analysis has been used extensively for parasitic optimization in [19]. The methods which use perturbation with a linear approximation to calculate sensitivities and optimize the parasitics are computationally expensive and not accurate, especially for application in a synthesis loop. As is evident from these applications, it is important to develop a technique that can obtain optimized parasitics efficiently and thus can be applied in the synthesis iteration process.

Symbolic sensitivity analysis is a formal method to determine the sensitivity of a circuit in terms of symbolic circuit parameters and independent variables like frequency which can be used for this purpose. In this chapter we propose how to use symbolic sensitivity analysis to optimize parasitic effects in the analog synthesis loop. The proposed method is introduced in the layout-aware analog circuit synthesis framework to reduce the overall runtime with reasonable accuracy. Parasitic modeling is achieved by using parameterized procedural layout generators. Parasitic effects inclusive performance model is obtained from ECDs [37], parasitic optimizing is obtained with a bottom-up differentiation of each coefficient graph [42].

Traditionally, the circuit synthesis step is followed by layout synthesis and each step is carried out independent of the other [38] [2]. This is followed by a verification step to check whether the desired performance specifications have been met after the layout generation and extraction. These steps are repeated till the desired performance specifications are met. This approach is extremely time-consuming and no structured feedback from previous runs can be readily used to re-design
the circuit if the layout fails to meet performance goals. In this chapter we present layout-aware synthesis based on optimized parasitics without performing layout generation as shown in Figure 1(b). Our method builds parasitic macro-models by analyzing several instances of the layouts. These macro-models are used in the synthesis loop to yield fast and accurate parasitic estimates. Parasitic optimization is done outside the synthesis loop which can save significant runtime. This flow is capable of achieving designs with layout closure.

![Flowchart image](image)

Figure 6.1: (a) Layout-in-loop Approach (b) Proposed Approach

We perform layout-aware synthesis based on accurate estimation of parasitic effects without performing layout generation. Instead of using them in the synthesis loop, we build parasitic macro-models by analyzing several instances of the layouts produced Module Specification Language (MSL) [18]. These macro-models are used in the synthesis loop to yield fast and accurate parasitic estimates. The performance and efficiency of the proposed algorithm have been demonstrated with benchmark circuits.
The rest of this chapter is organized as follows. Parasitic modeling techniques are described in Section 6.2. Section 6.3 presents symbolic sensitivity based parasitic optimization methodology. Section 6.4 provides a brief overview of parasitic optimization implementation in layout-aware synthesis loop. Experimental results are presented in Section 6.5 followed by conclusions in Section 6.6.

6.2 Parasitic Modeling

As component sizes vary during each synthesis loop, changes of design parameters during optimization may cause significant deviations in the layout parasitics.

In this section we introduce techniques to model parasitics introduced by layout. Usually there are two methods to obtain parasitics: one is physical layout extraction; the other is parasitic model estimation. Those two parasitic modeling techniques are introduced in the following two subsections.

6.2.1 Physical layout extraction technique

In physical layout extraction technique, we build parasitic macro-models by generating a number of sample layouts of the entire circuit, examine which area and coupling capacitances are extracted from these layouts and symbolically include only those capacitances in the sensitivity optimization model generation process. We use sensitivity based parasitic optimization method to determine the dominant parasitics and those macro-models generated are used in the synthesis loop to yield fast parasitic estimates with reasonable accuracy. The advantage of using physical layout extraction technique is that only relevant parasitics, i.e., those which have appeared due to extraction of layouts, are taken into account. This method may not generate all possible parasitic elements and
this may cause inaccuracy if a new physical capacitor is extracted. There is an additional overhead of generating the layout samples [28].

6.2.2 Parasitic model estimation technique

A procedure layout generation environment called MSL is used to specify layout generators. MSL is an attribute based high level language which allows hierarchical composition of layout primitives and permits their relative placement. A MSL program specifies a parameterized layout generator. When the parameter values, typically device sizes, are supplied, a concrete layout is obtained. Off-the-shelf extractors can be used to extract a circuit including parasitic elements from this layout [1].

As component sizes vary during synthesis, the layout geometry varies between iterations. This variation may generate varying sets of parasitic elements (resistances and capacitances) in each iteration.

MSL is a module generation language which generates a layout using a predefined layout template. In this chapter, we just consider parasitic capacitances. We have classified the parasitic capacitances into two categories for the purpose of modeling [1]:

**Intra-module parasitics**, which are further sub-classified into (a) diffusion capacitances and (b) non-diffusion capacitances which include well, gate and interconnect capacitances (both area and coupling capacitances are included).

In a MSL system a module has fixed templates. A fixed template always generates the same set of capacitances for any size. The set of these potential capacitances can be determined either by applying the knowledge of extraction rules or producing the layout of any one instance of the module. In MSL, once the template information for all modules is available, the set of all possible capacitances can be determined for the entire layout.

**Inter-module** also called interconnect parasitics, which include the area and coupling capacitances
of the inter-module interconnect fabric. The inter-module wire parasitics (area as well as coupling) are dependent on the placement and routing of the modules in the circuit. In MSL, the relative placement of the modules is fixed and the routing between modules is attained using routing boxes [18]. In this case, the complete set of area capacitances and intermodule coupling capacitances for the interconnects is found and determined. The advantages of using this method are that all the relevant capacitances are obtained without generating sample layouts. Some unnecessary capacitances are generated because of the exhaustive technique used for interconnect coupling capacitances. The limitation of this method is that it is valid for a fixed layout template methodology. The detailed description can be find in [28].

Module Characterization Table (MCT) [1] is a lookup table with one or more input variables (input columns) and one or more output variables (output columns). Input variables are usually the sizes of module parameters and output variables are the various capacitances or the diffusion area/perimeter values observed from the extracted circuits. Using MCT, the values of capacitances can be determined when the sizes of module parameters are known. The detailed description about MCT can be found in [1].

With the careful estimation of layout parasitics for both intra-module and inter-module, the macro parasitic model is built.

### 6.3 Symbolic Sensitivity Based Parasitic Optimization

Gradient is essentially the first order tendency of a function with respect to certain parameters. The gradient of the performance function with respect to design parameters is the performance sensitivity. In other words, sensitivity of an analog circuit is the mathematical measurement of variations in the performance metrics due to small perturbations of circuit parameter values, which plays a very important role in determining the critical design variables. The use of sensitivity
information is crucial to optimization, as it is simple to compute and guide the desirable direction which parasitic is critical. In this section, we present procedures to compute the performance sensitivity with respect to layout parasitics introduced by design parameters.

An ECD is a multi-root version of the DDD (a compact graphical representation of symbolic determinants [30]) for representing symbolic transfer functions. Each coefficient in an ECD of the numerator or denominator is represented by one root and has a corresponding coefficient graph. Every edge in the graph representation is weighted with a single circuit parameter, source or constant. The value of the initial vertex is the sum of products of the weights (circuit parameter) of all the outgoing edges and their corresponding terminal vertices. After the accurate parasitic estimation, ECDs including parasitics are generated. The differentiation algorithm is applied to each vertex of each coefficient graph in the ECDs to determine the dominant parasitics.

As an illustrative example, the ECD of a RC network without parasitic effects is shown in Figure 6.2 and with parasitic effects is shown in Figure 6.3. The ECD graphs of RC network without considering parasitic effects (Figure 6.2) is shown in Figure 6.4. As the comparison, the ECD graphs of RC network with parasitic modeling (Figure 6.3) is shown in Figure 6.5. In Figure 6.3 and 6.5 $cp1$, $cp2$, $cpv$ are parasitic effects included in synthesis loop for accurate performance estimation.

In order to evaluate the dominant parasitics in this ECD, we need calculate the circuit performance sensitivities with respect to each parasitic, and then include the critical parasitics in our synthesis loop for fast estimation. In this proposed method we consider each parasitic as the circuit parameter to evaluate its sensitivity value. The detailed procedures about how to calculate performance sensitivities has been demonstrated in [42] [41]. In our proposed method parasitic sensitivity and optimization are calculated and evaluated outside the circuit synthesis loop.
6.4 Analog Circuit Synthesis Implementation With Proposed Approach

The proposed circuit synthesis environment with parasitic optimized model is shown in Figure 6.6. The layout generation and extraction step is removed from the synthesis loop. The layout effects are taken into account and optimized. The output of the parasitic estimator is a netlist which closely resembles an extracted netlist. Circuit topology and search range for each parameter value of the
circuit are the inputs to the optimization engine. In this case simulated annealing is the core algorithm used for optimization which generates a new set of parameter values. Optimization engine then passes those new set of parameter values to NGSPICE (or other simulator) to perform operating point (OP) analysis and the small-signal values for all active devices generated by OP analysis are passed to the performance evaluator. The performance evaluator processes the simulation data to obtain the values of the desired performance parameters. These parameters are passed to the cost function evaluator to evaluate the cost which is used by the optimization engine to determine the new device size in each iteration process of optimization. This process is repeated until the desired performance goals are achieved.

6.5 Experiments

In this chapter, models and methodologies were developed for estimating and optimizing various parasitic capacitances. Synthesis experiments were performed on our five benchmark circuits. Single-Ended Op-amp1(SEO1) is the device model of CMOS Operational Amplifier in [21].
Figure 6.5: ECD of RC with Parasitic Effects

Two-stage Op-amp2 (TSO2) is borrowed from page 308 of [29]. Low Pass Filter (LPF), has been borrowed from [25]. LPF is a fourth-order Butterworth filter of Sallen-Key implementation. SEO2 is also a single-ended op-amp and TSO1 is yet another two-stage op-amp. SEO1 is used to implement the LPF. Because of the limited space we just show three of benchmark circuits for references in Figure 6.7.

Table 6.1 gives an account of the number of devices, nodes, parasitic capacitances, ECD depth, ECD vertices and edges included in the benchmark circuits. Table 6.2 presents the circuit performance comparison between the circuit with parasitic modeling and the one with optimized parasitic modeling. The maximum deviation is less than 1%. This results show that our parasitic
optimization method is acceptable with reasonable accuracy.

Table 6.3 presents the speedup of runtime results between traditional method and proposed methodology. In Table 6.3, WP stands for runtime of circuit synthesis with the complete parasitic modeling and PO is the proposed synthesis with optimized parasitic modeling. Parasitic optimization in circuit synthesis reduces number of parasitics and hence the runtime. From Table 6.3 we can see that the average iteration speedup for three benchmark circuits is around 22.4% because of the application of symbolic parasitic optimization analysis.

All the experiments mentioned above have been conducted on SunBlade 1000 with Solaris (SunOS), 2048MB RAM and 2-750MHz processors. The experimental results show a considerable speedup in analog synthesis with sensitivity-based parasitic optimization analysis.
6.6 Conclusions

Some performance parameters are more sensitive to certain parasitics as compared to others. Therefore, the sensitivity of performance parameters with respect to parasitics can be used to determine the crucial parasitics, thus the insignificant parasites can be ignored. This chapter has proposed a new methodology for parasitic optimization in layout-aware synthesis. The proposed method uses symbolic sensitivity analysis to implement fast parasitic optimization which speeds
Table 6.1: Parasitic Capacitances of Benchmark Circuits

<table>
<thead>
<tr>
<th>Circuit Name</th>
<th>Devices(#)</th>
<th>Nodes(#)</th>
<th>Parasitic Caps(#)</th>
<th>ECD Depth(#)</th>
<th>ECD-Vertices(#)</th>
<th>ECD-Edges(#)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSO1</td>
<td>5</td>
<td>11</td>
<td>31</td>
<td>11</td>
<td>668</td>
<td>4234</td>
</tr>
<tr>
<td>TSO2</td>
<td>9</td>
<td>11</td>
<td>30</td>
<td>11</td>
<td>1,024</td>
<td>8,479</td>
</tr>
<tr>
<td>SEO1</td>
<td>9</td>
<td>10</td>
<td>32</td>
<td>10</td>
<td>503</td>
<td>3,972</td>
</tr>
<tr>
<td>SEO2</td>
<td>16</td>
<td>17</td>
<td>56</td>
<td>17</td>
<td>1,661</td>
<td>13,522</td>
</tr>
<tr>
<td>LPF</td>
<td>22</td>
<td>22</td>
<td>48</td>
<td>48</td>
<td>11,583</td>
<td>71,563</td>
</tr>
</tbody>
</table>

up the synthesis process significantly with reasonable accuracy. The parasitic evaluation and optimization were removed from the synthesis process which reduces the time spent in synthesis.
## Table 6.2: Layout-Aware Circuit Performance Comparison

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Attribute Constraints</th>
<th>Parasitics</th>
<th>Optimized</th>
<th>% Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSO1</td>
<td>Gain≥ 40</td>
<td>41.1</td>
<td>41.1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>F_{3db}≥60</td>
<td>62.2</td>
<td>62.2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>UGF≥320</td>
<td>331</td>
<td>331</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PM≥50</td>
<td>53.6</td>
<td>53.6</td>
<td>0</td>
</tr>
<tr>
<td>TSO2</td>
<td>Gain≥ 23</td>
<td>25.1</td>
<td>25.1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>F_{3db}≥60</td>
<td>63.3</td>
<td>63.3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>UGF≥300</td>
<td>311</td>
<td>311</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PM≥50</td>
<td>51.2</td>
<td>51.2</td>
<td>0</td>
</tr>
<tr>
<td>SEO1</td>
<td>Gain≥ 50</td>
<td>52.1</td>
<td>52.1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>F_{3db}≥8</td>
<td>8.52</td>
<td>8.52</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>UGF≥200</td>
<td>221</td>
<td>219</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>PM≥60</td>
<td>51.2</td>
<td>51.2</td>
<td>0</td>
</tr>
<tr>
<td>SEO2</td>
<td>Gain≥ 35</td>
<td>35.2</td>
<td>35.4</td>
<td>0.57</td>
</tr>
<tr>
<td></td>
<td>F_{3db}≥20</td>
<td>20.3</td>
<td>20.5</td>
<td>0.99</td>
</tr>
<tr>
<td></td>
<td>UGF≥100</td>
<td>113</td>
<td>115</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td>PM≥60</td>
<td>61.2</td>
<td>60.9</td>
<td>0.5</td>
</tr>
<tr>
<td>LPF</td>
<td>Gain≥ 10</td>
<td>11.1</td>
<td>11.4</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td>F_{3db}≥950</td>
<td>960.3</td>
<td>965.2</td>
<td>0.51</td>
</tr>
</tbody>
</table>
Table 6.3: Iteration and Runtime Comparison Using Parasitic Modeling between with and without Optimized Parasitics Method

<table>
<thead>
<tr>
<th>Circuit Name</th>
<th>Performance Specification</th>
<th>Runtime(s) WP</th>
<th>PO</th>
<th>Speedup(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSO1</td>
<td>gain ≥ 40</td>
<td>410.1</td>
<td>329.3</td>
<td>19.7</td>
</tr>
<tr>
<td>TSO2</td>
<td>gain ≥ 50</td>
<td>636.2</td>
<td>481.6</td>
<td>24.3</td>
</tr>
<tr>
<td>SEO1</td>
<td>gain ≥ 45</td>
<td>470.1</td>
<td>362.5</td>
<td>22.9</td>
</tr>
<tr>
<td>SEO2</td>
<td>gain ≥ 45</td>
<td>770.1</td>
<td>624.6</td>
<td>18.9</td>
</tr>
<tr>
<td>LPF</td>
<td>gain ≥ 40</td>
<td>1170.5</td>
<td>885.4</td>
<td>24.4</td>
</tr>
</tbody>
</table>
Chapter 7

Conclusions and Future Work

Analog circuit synthesis is critical in today’s and tomorrow’s ever-increasing microelectronics market, in particular the markets for application-specific ICs and application-specific standard parts. Developing efficient tools which quickly yield robust analog designs is in great demand. Our research focuses on rapid determination of critical parameters in analog circuit synthesis loop. Critical parameters give valuable guidance for designers in determining which circuit parameter to be chosen for tuning and how much its value is changed. In this dissertation the use of fast symbolic sensitivity analysis techniques has been proposed and studied. The experiments have shown the accuracy and efficiency of the proposed algorithms. In this chapter we discussed the contributions made in this dissertation and presented some possible directions for the remaining work.

7.1 Contributions

Efficient sensitivity analysis based on ECDs

- Element-Coefficient Diagrams (ECDs) have been used for generating performance sensitiv-
ities with respect to circuit parameters. SECDs have been converted to C++ and compiled to calculate performance sensitivities repeatedly during the synthesis loop. The accuracy and efficiency of SECDs in comparison to the numerical simulator SPECTRE has been studied and observed.

**Accurate MOSFET sensitivity modeling and calculations**

- Accurate model for MOSFET sensitivity calculations based on ECD and small-signal model have been developed. Multiparameter sensitivity analysis based on small-signal model is more accurate than any other transistor sensitivity analysis methods reported so far in literature.

- The proposed methodology has been applied for the synthesis of a two stage opamp circuit. The speed and convergence of analog synthesis have been demonstrated.

**Sensitivity weight optimization algorithm and application**

- Techniques for the use of sensitivity weight optimization algorithm and symbolic performance evaluation in synthesis loop.

- Techniques for performance evaluation and sensitivity analysis at the same time.

- The efficiency of the sensitivity weight optimization algorithm and performance evaluation inclusion techniques has been demonstrated by comparisons to traditional synthesis method.

**Parasitic optimization and application based on symbolic sensitivity analysis**

- Techniques for fast parasitic optimization with symbolic sensitivity analysis, the deviation of optimized parasitic model is tiny.

- Apply parasitic optimization technique in layout-aware analog synthesis, the runtime speeds up considerably.
7.2 Future Directions

In this section, we present some possible directions for the remaining work for this dissertation.

Symbolic poles/zeros estimation and verification based on ECDs

Poles and zeros determine the frequency characteristics of the circuit. Poles are frequencies at which the transfer function is infinity. Zeros are frequencies at which the transfer function reaches 0. Symbolic expressions of poles and zeros of circuit transfer function can help circuit designers gain the insight on the circuit frequency-domain behavior and stability. The cost of finding the poles and zeros of a circuit has traditionally been high. Symbolic methods can be impractical for even moderate sized circuits, and even numerical methods are costly for modern VLSI problems.

It’s unknown how a symbolic expression can be derived for an arbitrary pole or zero in the transfer functions. However, if a pole or zero is well separated from the others, an approximated symbolic express can be derived from the s-expanded transfer functions by root splitting [31].

Consider the following polynomial:

\[ f(s) = a_0 s^n + \ldots + a_{n-1} s + a_n = 0 \]  \hspace{1cm} (7.1)

As explained in preceding chapter, where \(a_0, \ldots, a_n\) are numerators or denominators in ECDs.

Under the assumption that the root \(s_k\) is well separated from the other roots, i.e.,

\[ |s_1| \leq \ldots \leq |s_{k-1}| \leq |s_k| \leq |s_{k+1}| \leq \ldots \leq |s_n| \]  \hspace{1cm} (7.2)

Then, the root \(s_k\) can be approximated as

\[ s_k = -a_{n-k+1}/a_{n-k} \]  \hspace{1cm} (7.3)

An ECD is a multi-root version of the DDD for representing symbolic transfer functions. Each root in an ECD represents a coefficient of the numerator or denominator. It’s not hard to calculate
$a_{n-k+1}/a_{n-k}$. The expressions of poles and zeros can be approximated as the ratios of the symbolic coefficients of two consecutive power of $s$. We propose to extract poles and zeros based on this method and will compare the results with SPCTRE or Matlab.

**Symbolic pole/zero sensitivity calculation and verification**

Pole and zero sensitivities are interesting in a wide range of problems. Filter applications depends heavily on pole and zero location. Control theory also deals extensively with poles and zeros, especially with regard to stability, system response and compensation. In addition, many analog design techniques are based on frequency-domain models, e.g., the single dominant pole model for operational amplifiers [15]. In analog circuit design (especially for analog filter design), one is often more concerned with the sensitivity of zeros or poles for a circuit transfer function instead of the normalized function sensitivity. A first order indication of how the poles and zeros shift when circuit parameters are perturbed is provided by sensitivities, and is of great value to designers. Based on the results of poles and zeros estimation, we propose a way to derive the sensitivity of any or specific zero or pole with respect to a circuit element $p_j$ symbolically. From the Equation 7.3 and the definition of the sensitivity, we derive:

$$S_{p_j}^{ZP} = \frac{\partial s_k}{\partial p_j} = -\frac{\partial a_{n-k+1}}{\partial p_j} = -\frac{1}{a_{n-k}} \frac{\partial a_{n-k+1}}{p_j} - \frac{a_{n-k+1}}{a_{n-k}^2} \frac{\partial a_{n-k}}{\partial p_j}$$

(7.4)

where $ZP$ represents any pole or zero.

From equation 7.4 it is evident that the major task in computing the first order derivative of poles or zeros is to compute the first order derivative of relevant symbolic coefficients of the numerators or denominators. Element-coefficient diagrams store each coefficient of the numerator and denominator as separate coefficient graphs, while sharing common sub-graphs. Due to this property of ECDs it is straightforward to obtain the derivative of each symbolic coefficient. There
are detailed description about how to obtain the derivative of each symbolic coefficient in Chapter 3.3.

The method above is under the assumption that the root $s_k$ is well separated from the other root which is not universal. There is another method which can acquire pole/zero sensitivity accurately under any circumstances.

A symbolic circuit transfer function is in this general form:

$$H(s, p) = \frac{N(s)}{D(s)} = \frac{\sum_{i=0}^{m} f_i(p) s^i}{\sum_{i=0}^{n} g_i(p) s^i} \quad (7.5)$$

where $s$ is the frequency Laplace variable and $p$ denotes the vector of all circuit parameters. The coefficients of the numerator and denominator are represented by $f_i$ and $g_i$ respectively.

It is possible to derive the following relationship:

$$\frac{\partial H}{\partial ZP} = \frac{1}{D^2} \left( \frac{\partial N}{\partial ZP} D - \frac{\partial D}{\partial ZP} N \right) = \frac{1}{D^2} \left( D \sum_{i=1}^{m} i * f_i * z p^{i-1} - N \sum_{i=1}^{n} i * g_i * Z P^{i-1} \right) \quad (7.6)$$

Thus, the pole/zero sensitivity $H_{zp}^{ZP}$ can be derived:

$$\frac{\partial ZP}{\partial p_j} = -\left( \frac{\partial H}{\partial p_j} \right) / \left( \frac{\partial H}{\partial ZP} \right) \quad (7.7)$$

We have illustrated the algorithm about how to calculate $\frac{\partial H}{\partial p_j}$ in Chapter 3, $\frac{\partial H}{\partial ZP}$ can be acquired from Equation 7.6, so the pole/zero sensitivity can be derived easily.

The two methods above provided pole/zero sensitivity with respect to circuit parameters which give valuable guidance regarding which circuit parameter(s) is critical to circuit performances. There is no commercial tool to do sensitivity calculations, we will manually calculate the any pole/zero sensitivity value by Matlab and compare the result with our proposed method.

**Layout-in-Loop synthesis of radio-frequency low noise amplifiers using efficient parasitic-inclusive symbolic sensitivity analysis**
Circuit synthesis is the process of determining the numerical values for unsized circuit elements of a fixed circuit topology, by using a combinatorial optimization algorithm, in order to satisfy a set of performance constraints. This traditional process when applied to radio-frequency (RF) circuits, suffers from two significant shortcomings. First, the parasitics introduced in the layout phase are not considered during optimization. As the performance of an RF circuit is extremely sensitive to these parasitics, it often leads to the failure of the sized circuit after layout. The second drawback is the extremely expensive computations for performance estimation because of using numerical analysis solely in the synthesis loop.

We propose to adopt layout-in-loop parasitics to alleviate the problem of performance degradation due to layout parasitics. The problem of computationally intensive numerical simulations can be avoided to an extent by using circuit performance models for estimating the behavior of a particular design and including sensitivity techniques to speed up the layout-in-loop synthesis process.

**Techniques on Weight Optimization Algorithm**

Chapter 5 proposed a weight optimization algorithm. The core of sensitivity weight optimization methodology is to assign various sensitivity weight vectors in different annealing procedures based on the temperature. In high temperature phase, we assign large sensitivity weights on the devices with high sensitivity values and small weights on those devices with low sensitivity values. Sensitivity weight is a very critical factor for synthesis efficiency. In this dissertation, the sensitivity weight of each parameter is proportional to its sensitivity value. In order to improve the synthesis efficiency in further step, techniques of adaptive weight adjustment will have to be developed.
7.3 Summary

Sensitivity is an efficient way to speedup analog synthesis. But we feel that symbolic sensitivity analysis is not mature enough at this time for a general modeling tool to be developed. We must rely on manual analysis of circuits to establish an appropriately constrained design space. Thus a fully automatic methodology for the development of accurate and efficient sensitivity application is still an open research problem.
Bibliography


