PERFORMANCE-DRIVEN PARASITIC-AWARE LAYOUT RETARGETING AND OPTIMIZATION FOR ANALOG AND RF INTEGRATED CIRCUITS

ZHENG LIU

Performance-Driven Parasitic-Aware Layout Retargeting and Optimization for Analog and RF Integrated Circuits

by

© Zheng Liu

A Dissertation submitted to the

School of Graduate Studies

in partial fulfillment of the requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

December 2010

St. John's Newfoundland

ABSTRACT

Performance of analog and radio-frequency (RF) integrated circuits is highly sensitive to layout parasitics. Layout-induced parasitics must be optimized to achieve desired circuit performance. This dissertation surveys the previous analog design automation approaches and presents an improved performance-constrained algorithm that can automatically conduct template-based parasitic-aware retargeting and optimization for analog and RF layouts. Piecewise sensitivities are deployed to represent the dependence of performance with respect to layout parasitics. The algorithm then uses these piecewise sensitivities to control parasitic-related layout geometries by directly constructing a set of performance constraints, subject to the maximum allowed performance deviation. Different from previous approaches that only consider parasitic resistances and wire-substrate capacitances, parasitic inductances and wire-coupling capacitances are taken into account to enable successful layout retargeting, in particular when handling RF layouts. The formulated problem is solved using graph-based techniques, combined with mixed-integer nonlinear programming (MINLP). The proposed method is incorporated into a template-based layout design tool called IPRAIL. The proposed algorithm has been demonstrated to be effective and efficient for generating target analog/RF layouts during process migration and/or performance retargeting.

ACKNOWLEDGEMENTS

First of all, I would like to take this opportunity to extend my gratitude and great appreciation to Professor Lihong Zhang, who has supervised and supported me constantly during the master's program. Dr. Zhang provided me with a great chance to work with him and fostered my academic success. I learned a lot from his knowledge and scientific spirit.

I would also like to thank my thesis examiners, Professor Howard Heys and Professor Theodore Norvell, for their time and patience. In addition, I would like to thank Mr. Nolan White, for his generous and patient technical support.

I am very thankful for my beloved father, mother, grandmother and my wife. They give me inexhaustible love, patience and understanding without any complaints.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

SoC – System on Chip

IC – Integrated Circuit

RF – Radio Frequency

CAD – Computer Aided Design

DRC – Design Rule Check

IP – Intellectual Property

RC – Resistance and Capacitance

RLC – Resistance, Inductance and Capacitance

MINLP – Mixed Integer Nonlinear Programming

LP – Linear Programming

NLP – Nonlinear Programming

AC – Alternating Current

ASIC – Application Specific Integrated Circuit

ECO – Engineering Change Order

Opamp – Operational Amplifier

LNA – Low Noise Amplifier

GUI – Graphic User Interface

NF – Noise Figure

IIP3 – The 3rd Order Intercept Point

PB-RC – The Template-Based Parasitic-Aware Layout Retargeting/Optimization Method

CIF – Caltech Intermediate Format

P-LP – The Proposed Parasitic-Solution Generation Flow

P-UC – UC-Berkeley Sensitivity Computation Techniques

PMI – The Proposed Performance-Constrained Mixed-Integer Retargeting Method

PB – Bound-Based Parasitic-Aware Retargeting Method

PS – Retargeting Using Single Worst-Case Sensitivities (Non-Piecewise)

VLSI – Very Large Scaled Integration

PM-R – Retargeting Considering Parasitic Resistances

PM-RCC – Retargeting Considering Parasitic Resistances, Wire-Substrate Capacitances and Wire-Coupling Capacitances

PM-RC – Retargeting Considering Parasitic Resistances and Wire-Substrate Capacitances

PM-RLC – MINLP Retargeting Considering Parasitic Resistances, Wire-Substrate Capacitances, Wire-coupling Capacitances, Self and Mutual Inductances

1. Introduction

The demand for smaller, cheaper, more portable electronics has been significantly increased in wireless communication and consumer electronics. This demand motivates the semiconductor industry to move towards the technology of systems-on-chip (SoC). The SoC applications accelerate the growth of mixed-signal integrated circuits (ICs). Thus, multiple functional sub-blocks including digital, analog, and radio frequency (RF) circuits, which used to reside in separate chips, can be integrated into one chip. The analog portion in mixed-signal designs is inevitable due to the nature of continuous signals in the external environment. Powerful CAD tools and cell-based design methodologies have been significantly advanced to facilitate design automation for digital circuits. However, up to now, analog circuit designers still have to spend an extremely large amount of time and disproportionate effort in conducting simulations and nontrivial layout design due to insufficient support from analog automation tools.

1.1 Background

For mixed-signal IC designs, even though the analog potion may occupy a very small fraction of the total chip area, it is often responsible for design errors and overspending issues such as delayed time-to-market or extra design cost. With traditional analog design methods, designers experience a tedious and error-prone design process to ensure tradeoffs among the major design goals such as gain, bandwidth, noise reduction,

stability, linearity and power minimization. These handcrafted design methods are largely dependent on the high-level expertise of experienced analog designers. Due to the complexity of analog performances with respect to layout geometries, layout-induced performance degradation is more significant with the advancing process technologies. The performance functions with respect to layout parameters (e.g., interconnect parasitics) are more difficult to be optimized compared to the case of digital designs. Moreover, radio frequency (RF) is advocated due to the large demand of high-speed IC products. At radio frequency, parasitic inductive impacts and wire-coupling capacitive effects become very significant in affecting the layout performance, besides the resistive/capacitive parasitics considered for lower-frequency analog designs. Thus, the performance functions with respect to layout-induced parasitics are more difficult to be modeled in the RF domain. Using the current analog/RF design tools, a market-ready IC design typically requires unsystematic exhausting iterations among constraint editing, performance optimization and layout generation, since an initial optimization usually brings DRC (i.e., design-rule-check) errors or performance failures. As a whole, analog/RF automation has become a design bottleneck for the growing mixed-signal SoC market.

Recently, significant progress has been made in analog optimization methodologies, which fall into two categories: macro-cell based physical synthesis (i.e., device-generation, placement and routing) and template-based layout retargeting. For example, [1][2][3] introduced several analog tools that automatically synthesize analog circuits while meeting desired performance specifications. However, these tools only considered

the device sizes and biasing as factors that affect circuit performance (e.g., no consideration on parasitics or layout structures), which is insufficient for high-performance analog layout design.

Moreover, since semiconductor manufacturers continue to update technologies towards even smaller transistor feature sizes (e.g., from 0.18um to 0.13um, from 0.13um to 90nm, etc.), people have to migrate existing mixed-signal layouts in an original technology process to the ones in a new technology process [4]. Due to the differences in technology properties and insufficient analog CAD tools, the process of migration typically means a tedious re-design of new layouts. For digital layouts, intellectual property (IP) reuse can be readily realized by simply adopting the scalable cell libraries and available digital placement and routing tools to migrate the existing high-level VHDL or Verilog designs to target designs in a new process. However, analog designers have to experience a full cycle of redesigning from scratch since analog IP reuse is not available. In order to speed up the re-design process, powerful computer-aided-design (CAD) tools are needed for effective and efficient technology migration of analog layouts.

Besides technology migration, analog/RF designers usually intend to upgrade an old design to gain better functionalities (i.e., performance retargeting). To avoid a complete re-design of layouts, a template-based retargeting algorithm [4] was proposed as a solution to reuse the expertise extracted from an input layout, during the target-layout compaction. In this methodology, a moderate number of performance-related constraints (e.g., constraints related to interconnect parasitics, device matching and symmetry) are added into an extracted template to form an updated template. A target layout can be

quickly generated after solving the updated template. However, the template-based retargeting method as proposed in [4] fails to take into account the performance degradations due to layout parasitics.

1.2 Contributions

It is worth mentioning that a successful analog/RF layout design methodology typically features satisfactory performance goals, efficient solving, accurate and global parasitic control, as well as high-level design automation. This dissertation introduced a performance-driven template-based layout retargeting algorithm for analog and RF layouts. Compared to some previous methodologies, the major contributions of this dissertation are:

1. A solution of RF layout retargeting is designed by applying a lumped RLC interconnect model, and incorporating self inductances, mutual inductances and wire-coupling capacitances into the formulation [5];

2. Accurate performance sensitivities due to parasitics are applied by adopting a piecewise-sensitivity model [6];

3. Global performance optimization is achieved by replacing the imposed parasitic bounds with the proposed performance constraints [7] [8];

4. A mixed-integer nonlinear formulation is constructed for the parasitic-aware problem which is solved by a powerful MINLP solver;

5. Experimental results on several analog/RF designs verify the layout quality improvement and execution time reduction compared to the alternative methods.

1.3 Structure of the Dissertation

The rest of the dissertation is organized as follows. In Chapter 2, we present a literature review of prior analog/RF design-automation methodologies. In Chapter 3, the fundamentals of template-based layout retargeting are introduced as the preliminary work for the dissertation. A simulation-based algorithm for parasitic-solution generation is then discussed in Chapter 4. Chapter 5 introduces a performance-constrained parasitic-aware retargeting methodology for analog layouts. The RF retargeting method is presented in Chapter 6 followed by the conclusions and future research topics presented in Chapter 7. Moreover, a tutorial of the implemented layout tool (called IPRAIL) is given in Appendix I for layout retargeting of analog/RF layouts. Finally, the publications, as the outcome of the dissertation research, are listed in Appendix II.

2. Survey of Analog Layout Design Automation

This chapter surveys various methodologies and design tools for analog design automation. In order to better conduct the literature review, important terminology is defined. Then commercial and academic developments for analog design automation are discussed. Important layout issues of parasitic, symmetry and matching are also studied as a preparation for the later chapters.

2.1 Terminology

In order to better survey the previous work in the literature and explain the research in this dissertation, some important terminology is defined as follows.

- **Circuit Performance**

Electrical functionality of a circuit. It is used to evaluate the effectiveness of circuit operating functions (e.g., the AC Gain of an operational amplifier).

- **Constraints**

The requirements which the solution to an optimization problem is subject to meeting.

- **Interconnect Parasitic Models**

The models used to represent circuit interconnect. Within the models, components of parasitic resistances, capacitances and inductances are usually included.

- **IPRAIL**

A parasitic-aware automatic layout optimization and retargeting tool [4] [9]. The proposed algorithms in this thesis are incorporated into this tool. The proposed retargeting algorithms are implemented in C/C++ as modifications to this tool.

- **Layout Tile**

A rectangle on a layer within a layout. A net of a layout is composed of a set of layout tiles.

- **Layout Generation**

Layout generation is a process of solving a formulated layout optimization problem, and mapping back the solution geometries to construct a target layout.

- **Parasitic Upper Bound**

An upper-limit value of a certain parasitic that ensures the baseline of certain performance specification.

- **Performance Sensitivity**

The dependence of circuit performance with respect to circuit parameters (e.g., current and voltage) or layout parasitics.

- **Performance Deviation**

The variation of circuit performance due to factors such as temperature, operating frequency, or layout parasitics.

2.2 Literature Review

Analog design automation is of vital importance to mixed-signal integrated circuits and application specific integrated circuits (ASICs) [9]. Commercial computer-aided design (CAD) tools have been developed and widely used in the digital IC domain.

However, analog design is still largely handcrafted work, due to insufficient support from available commercial analog CAD tools. In nature, analog design is more difficult than the digital one due to the complex analog performance degradation. Analog design becomes even harder during process migration towards smaller transistor feature sizes. In the following, a survey of analog/RF layout automation is presented in Section 2.2.1, for the industrial development, and Section 2.2.2, for the academic development.

2.2.1 Industrial Development

Decades ago, mature digital CAD tools were developed and widely used in industry, but no commercial CAD tools were available for analog design automation. Thus, the analog layouts were designed in a traditional handcrafted manner. With the growing demand of larger-sized mixed-signal IC products, handcrafted analog design methods lead to unacceptable time-to-market. The analog portion usually accounts for the overspending issues (e.g., excessive design time) of mixed-signal IC designs.

In order to overcome the bottleneck of analog automation, commercial analog CAD tools have been developed recently by CAD companies such as *Cadence Design Systems Inc*, *Synopsys Inc*, *MentorGraphics Inc*, *SpringSoft Inc*, *etc.* The *Cadence Virtuoso* platform features full-custom IC design, which has been the most powerful digital and mixed-signal IC platform for industrial applications. To update the traditional Virtuoso platform with improved analog/RF capability, NeoCell and NeoCircuit (detailed in the latter part of this section) have been integrated into the overall Virtuoso IC design flow. Synopsys brought significant contributions to analog design automation, mainly in terms

of powerful analog simulators such as HSPICE [10] and HSPICERF [11]. Moreover, *Synopsys DesignWare®* [12] provided a set of industry-ready analog IPs such as *Analog-to-Digital Converters* (ADCs) and *Digital-to-Analog Converters* (DACs). To overcome the critical time-to-market issues due to more complex analog designs, *Mentor Graphics* offered an *IC Station®* layout suite [13] with a complete analog design flow. This tool suite is able to perform fast device generation, schematic-driven placement, automatic routing and post-layout verification. The key advantages of this methodology are the on-the-fly DRC leading to reduced time-to-market, and the capabilities of performing ECO (i.e., engineering change order). *SpringSoft* offered a custom layout system called *Laker* [14] that applies a constraint-driven layout generation flow and template-based device generation. Moreover, it features controllable automation, which allows designers to interact with *Laker* to control the design automation. Since *Cadence Virtuoso* is the most popular IC platform in industry, NeoCell and NeoCircuit that are integrated into this platform are detailed in the following.

NeoCell [15] and NeoCircuit [16] were developed in 1999 for achieving automatic analog/RF circuit sizing, performance optimization and layout synthesis. Recently, these tool suites have been integrated into the Cadence Virtuoso full-custom IC design platform. The Neo products have been widely used for analog/RF design automation in the mixed-signal IC industry for several years.

Virtuoso NeoCircuit is able to automatically resize and optimize analog and RF cell schematics using the designer's simulators of choice. NeoCircuit first captures the device relationships from a cell schematic. Based on the captured device information, multiple simulations, such as Transient and AC, can be conducted using the designer's simulators

of choice. Designers are then allowed to specify a set of performance goals (e.g., *AC gain*, *phase margin*, etc.). These performance goals will be enforced in the optimization. NeoCircuit features both global and local optimizations. By dictating the intrinsic relationship between design objectives and analog parameters (e.g., voltages, currents, resistance, and capacitance), a global optimization can be conducted to meet most of design goals while minimizing power. However, some goals may not be met after a global optimization since performance of analog/RF circuit is very sensitive to physical effects. To meet all performance goals, a local optimization can be conducted with a focus of improving the failed performances due to a global optimization. For most small-sized or medium-sized circuits, all design goals can be achieved after a local optimization. As a whole, NeoCircuit is able to optimize and size analog/RF cells to meet designer's performance goals. The optimized cell schematics by NeoCircuit can then be constructed as a library of reusable analog intellectual property (i.e., IP). To better illustrate the NeoCircuit design flow, an example of circuit optimization is given as shown in Figure 2-1 and Figure 2-2.

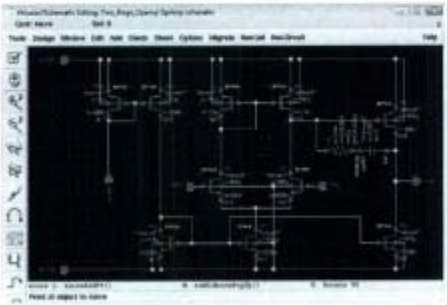
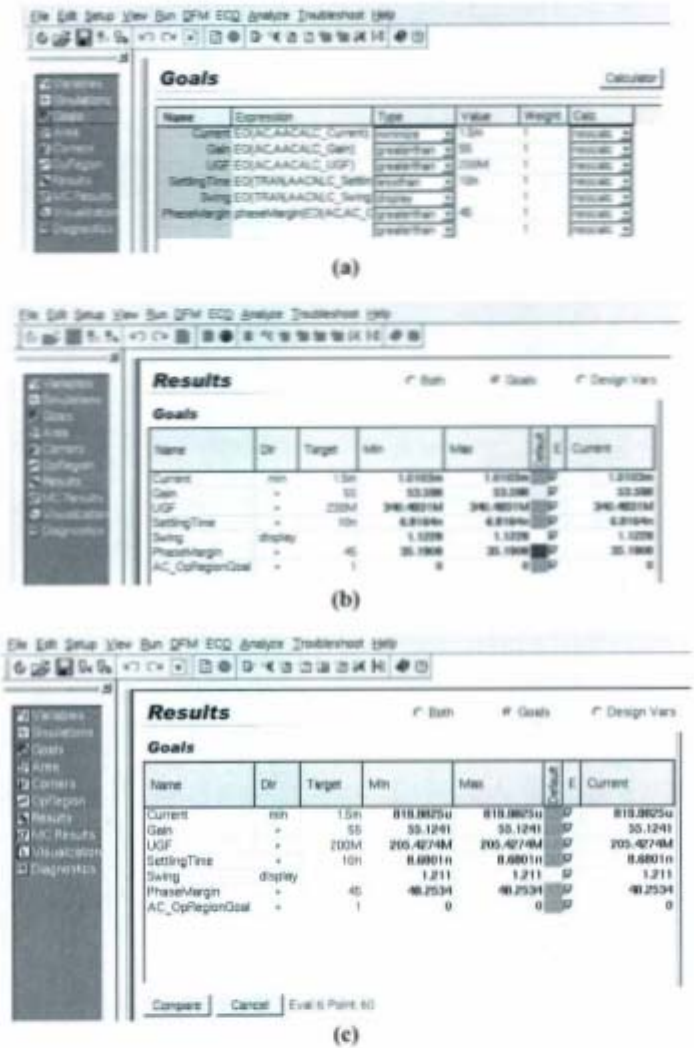


Figure 2-1 The cell schematic of a two-stage operational amplifier.



(a)

(b)

(c)

Figure 2-2 (a) Specifying performance goals using NeoCircuit, (b) reported performances after a global optimization, (c) reported performances after a local optimization.

A small two-stage opamp as shown in Figure 2-1 was optimized in a gpdk180 process technology. After the device-relationship extraction and simulations, designers can specify a set of performance goals as shown in Figure 2-2(a). For example, the specification is > *55 dB* for AC *gain* and > *45 dB* for *phase margin*. Figure 2-2(b) shows the summary of achieved performances after the global optimization. Most performance goals have green boxes that indicate these goals were met, while some performance goals (e.g., AC *gain* and *phase margin* in the graph) have yellow or red boxes indicating they were failed ones. To improve the failed AC *gain* and *phase margin*, a local optimization was conducted and the result is shown in Figure 2-2(c), where all performance goals were met.

Virtuoso NeoCell is able to translate a set of cell schematics into a full-custom optimized layout with user-imposed constraints. These constraints include wire style, device style, device matching, cell dimension, etc. With a set of user-imposed constraints, NeoCell can conduct physical-level synthesis including device generation, placement and routing in an automatic or interactive manner. Moreover, NeoCell is able to achieve process migration and performance retargeting by performing an ECO (i.e., Engineering Change Order) flow as shown in Figure 2-3.

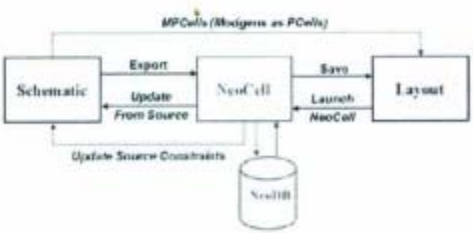
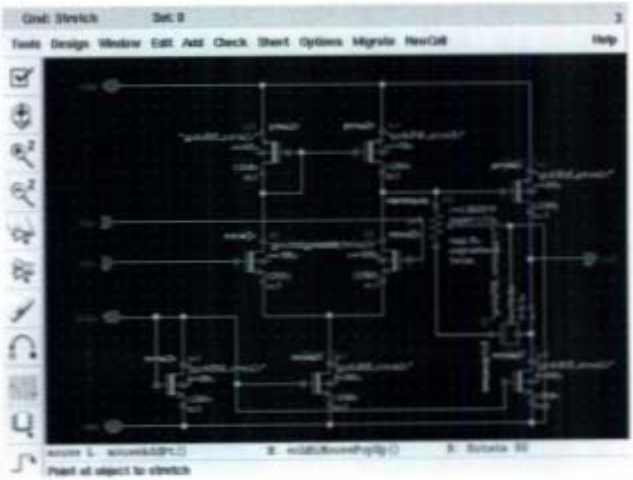


Figure 2-3 The ECO flow of NeoCell [15].

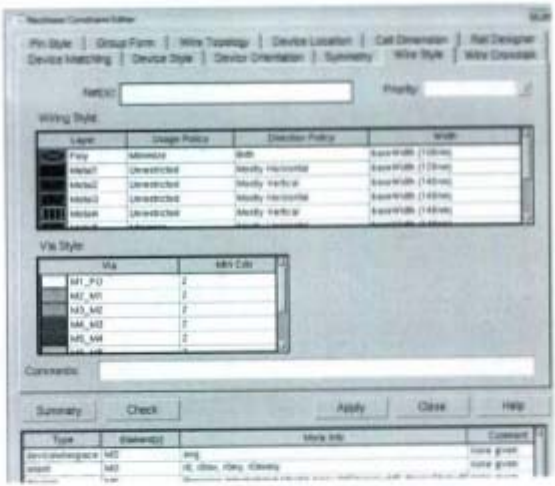
To better demonstrate the NeoCell design flow, an example cell schematic as shown in Figure 2-4(a) was translated into a full-custom optimized layout. After the module generation, designers can specify desired constraints using a constraint editor as shown in Figure 2-4(b). For example, for wiring-style constraints, the imposed usage policy is *minimizing* in Figure 2-4(b). With designer-imposed constraints, automatic placement and routing were conducted to generate a target layout. The generated layout after an automatic placement is shown in Figure 2-5(a) and the final layout after a routing is shown in Figure 2-5(b).

However, DRC (i.e., design rule check) and routing errors usually occur after an automatic layout synthesis. Moreover, some post-layout simulated performances may fail to meet their specifications after the initial iteration of automatic synthesis. Fixing these errors requires time-consuming iterations of re-module-generation, re-constraining, re-placement and re-routing.

Moreover, the ECO flow requires considerable iterations of re-constraining and re-layout during process migration or performance retargeting. DRC errors and post-layout performance loss usually occur during process migration or performance retargeting because the complete constraint set is partially enforced in several different phases rather than in a global manner. For example, the NeoCell module generation is parasitic-aware but the placement and routing are not. Thus, fixing the layout-induced performance loss due to an ECO requires time-consuming and error-prone iterations between topological and physical design phases, solely dependent on designer's experience.

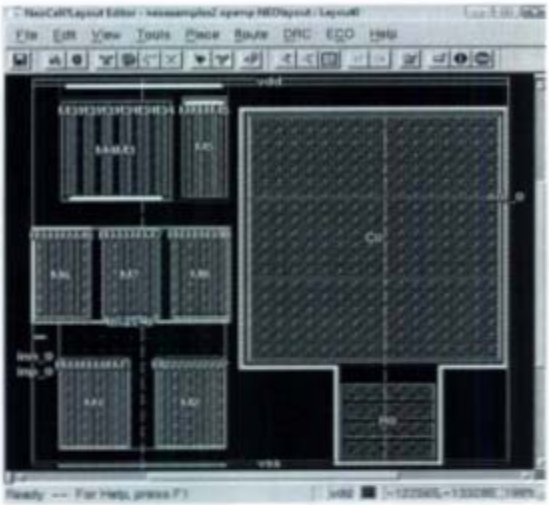


(a)

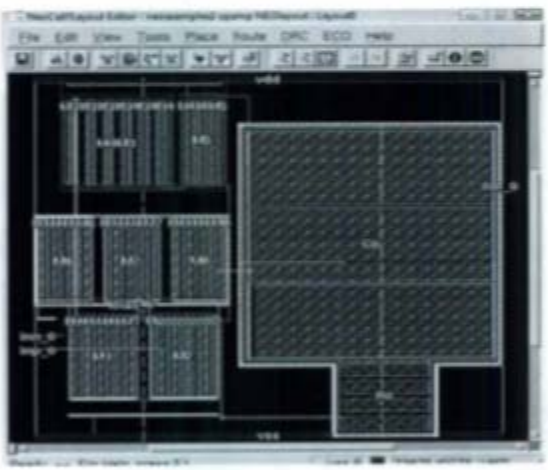


(b)

Figure 2-4 (a) A schematic of an opamp, (b) a GUI of NeoCircuit constraint editor.



(a)



(b)

Figure 2-5 (a) A layout generated after an automatic placement by NeoCell, (b) a layout after an automatic routing by NeoCell.

2.2.2 Academic Development

In the past decades, system-level analog compilers have been developed and they are promising in handling some specific circuit systems that are characterized by hierarchical structures. For instance, automatic synthesis of operational amplifiers (i.e., opamps hereafter) and comparators were introduced in [17][18], switched capacitor filters in [18] and data converters in [20]. These synthesis engines meet some design aspects of analog ICs, but their effectiveness is only verified for some specially characterized circuits, which sacrifices the input generality.

Several other CAD tools as introduced in [21][22][23] were developed to automate the generation of analog layouts. However, they sacrifice the expertise owned by experienced analog designers. The analog layout synthesis tool in [21] uses a top-down template-based design approach. This approach uses a fine-tuned layout as the input to quickly generate a target layout with good quality. However, for a different design, considerable coding must be re-conducted to generate another specific template, which actually results in excessive design effort. Meye Bexten *et al.* developed a rule-based analog layout system called *ALSYN* [22]. This system applies a rule set to control the quality of the generated layout but the flow greatly depends on context orders. *ILAC* is proposed in [23] as a process-independent CAD tool that automatically generates layouts for analog CMOS circuits. Simulated annealing and slicing structures are used to facilitate automatic placement in *ILAC*. However, this tool has difficulties in generating high-performance dense analog layouts, since it directly borrows some features from the digital layout styles and uses them in the analog field.

Malvasi *et al.* proposed a fully integrated constraint-driven analog layout system [24]. This system translates high-level performance specification into lower-level bounds on parasitics and geometric parameters. Then some specialized layout engines use these bounds to drive stack generation, placement and routing. However, the parasitic bounds derived from the specification are generated solely based on simulations without considering the particularity of layout geometries. Thus, the extracted bounds are not accurate and this straightforward translation may render viable parasitic optimizations over-constrained or even unsolvable. Beyond the basic functionality provided by *KOAN/ANAGRAM-II* [25] handling device-level layout automation, *LAYLA* [26] considering performance and manufacturability issues is a helpful extension in the analog layout design. This tool takes advantage of simple module generators and evaluates the predetermined sensitivities with respect to an intermediate layout solution in the layout generation process. Although it is able to handle all the possible geometry-sharing optimization during the placement, the optimal solutions in the module generation process are difficult to be obtained in the placement and routing stages. Moreover, those device-based placement and routing methodologies tend to have difficulties in reusing designers' expertise.

For reusing designer's expertise, a layout synthesis tool called *ALADIN* [27] was developed to incorporate designers' knowledge into the synthesis process. This tool applies a flexible strategy of module generation such that designers can build variable modules in a technology-independent manner. The design knowledge can then be simply represented in the generated layouts. Nevertheless, this tool can only handle small or

medium sized analog circuits, and complex performance and manufacturability constraints are not considered.

For the purpose of effective analog IP reuse, Jangkrajarng *et al.* proposed a template-based tool called IPRAIL (i.e., *Intellectual Property Reuse-based Analog IC Layout Automation*), which automatically retargets existing analog layouts for technology migration and/or updated specifications [4]. This method realizes analog IP reuse by extracting a symbolic template (i.e., a set of linear constraints) from an existing expertise-embedded layout. Unfortunately performance consideration is not included in that work. As a matter of fact, layout geometries of device matching and symmetry, interconnect parasitics, thermal, and substrate effects can significantly impact analog circuit performances [28][29]. Managing performance degradation is essential for the success of analog design automation.

Moreover, a layout-aware synthesis solution was introduced in [30] for analog cells. This method features minimization of the iterations between electrical synthesis and physical synthesis. It handles the parasitic-aware sizing and the geometry-constrained sizing in a global way. However, this approach has difficulties in a global compaction of large designs with many analog cells, since no general solutions of hierarchical decomposition and specification transmission for analog designs are available. A layout tool called *ALG*, which can conduct hierarchical module generation, placement and routing with the aid of two partitioning algorithms, was introduced in [31]. This tool takes advantage of performance sensitivities to control parasitic capacitive effects as well as parasitic mismatch in the layout synthesis. Unfortunately, this work failed to handle parasitic resistances and inductances, which are critical in degrading circuit performance

for high-performance analog/RF designs. Moreover, the advocated performance sensitivities are generated solely by the developed simulator called *YASA*, and no consideration is taken for the complexity of performance sensitivities for different ranges of a parasitic value (e.g., a sensitivity of AC *gain* is 1dB/pF when a parasitic capacitance is within 5 pF, but this sensitivity can become 3 dB/pF when the parasitic is above 5 pF).

Layout parasitics arise from transistor source and drain capacitances, interconnect resistances, wire-substrate capacitances, wire-coupling capacitances, and interconnect inductances. These parasitics significantly affect analog/RF circuit performances such as gain, bandwidth, phase margin, gain margin, etc. Thus, parasitic issues must be clearly addressed in the analog/RF layout generation process.

Zhang *et al.* further proposed a parasitic-aware optimization flow as an effective solution to analog layout retargeting [9]. In that work, an already fine-tuned layout is used to create a symbolic template involving floorplan, symmetry, and device/wiring-alignment information. Mathematically, the symbolic template is represented by a set of constraints on the target-layout geometry. Parasitic bounds, which can be used to constrain corresponding geometric expressions, are manually or semi-automatically estimated before conducting the layout generation. These bounds are then used to control related layout symmetries towards minimum layout area and satisfactory performance. The generation of the target layout, subject to the constraints in the symbolic template, is formulated and solved using graph techniques combined with nonlinear programming. However, the following disadvantages exist in that flow. First, inductive impacts or wire-coupling capacitive impacts are not involved in the parasitics optimization, which inevitably degrades its effectiveness for RF layout retargeting. Secondly, parasitic bounds

are enforced separately without considering their correlations/cancellations in affecting the same performance, which fails to achieve a global performance optimization. Thirdly, as a separate step, the parasitic bounds are generated before optimization. So the parasitic-related geometry limits may over-constrain the problem and render it unsolvable.

Complete consideration of resistive, inductive, and capacitive parasitics are essential to the success of parasitic-aware analog/RF design particularly at radio frequency [32][33]. The performance of RF designs is normally measured by S-parameters, noise figure (*NF*), and nonlinearity, mainly in terms of the 3rd order intercept point (*IIP3*). At lower frequencies, performance tends to be less sensitive with respect to parasitic inductances and wire-coupling capacitances, compared to the parasitic resistance counterparts. However, at higher frequencies, inductive and wire-coupling capacitive parasitics may cause significant signal degradation (e.g., signal delay, crosstalk, ringing, reflections and distortions) [33]. Due to the increasing operating frequency, the parasitic inductive and wire-coupling capacitive impacts of interconnects become more significant in affecting RF circuit performance. Thus, inductive parasitics (i.e., self and mutual inductances) and wire-coupling capacitances must be controlled to avoid malfunction of RF designs, if only considering parasitic resistances and substrate-capacitances otherwise.

2.3 Important Layout Issues

Analog/RF circuit performance is mostly dictated by layout geometric topology and device sizing [4]. These layout geometries are in the form of interconnect parasitics,

device/parasitic matching, device/parasitic symmetry, current density in interconnects, thermal effects, substrate effects [34] [35], etc. The focus of this dissertation is layout parasitics, symmetry, and matching.

2.3.1 Layout Parasitics

Interconnect parasitics can severely affect the performance of analog/RF layouts. For analog layouts, parasitic resistance and capacitance must be restricted within certain upper bounds to control their impact on circuit performance. Ignoring parasitic issues in analog layout design usually results in malfunction of analog IC products.

With advanced process technologies, the delay of transistors is decreased due to smaller transistor-feature-sizes. However, due to the growing density and increasingly complex multi-layer wiring structures, the interconnect delay does not follow this trend [36]. For instance, in 0.18 μm process technology, interconnect delay can occupy almost 70% of the path delay, and the physical capacitance due to interconnect capacitance can become dominant in power estimations. Moreover, mainstream submicron and nanometer processes (e.g., 180nm and 90nm) advocate complex multilayer metallization wiring structures with advanced dielectric materials [37]. The congested thin-and-long interconnects usually come with low voltage and fast-clocking edges. For mixed-signal designs, the wire-coupling capacitances between these long wires can bring severe noise and crosstalk. Thus, circuit simulation with variable parasitics is a must before a physical design phase to accurately analyze these effects. Moreover, early estimation of sensitive parasitic components is very significant to avoid relayout due to a parasitic-induced

performance loss. To achieve a global analog physical design, parasitic impacts represented by related layout geometries must be incorporated into the physical design formulation.

Moreover, the demand of high-speed IC products accelerates the growth of RF layouts. For designs at high frequency, parasitic inductive impact becomes a critical issue especially when interconnects have large currents [38]. Wire-coupling capacitances are increased dramatically due to the advocated dense wiring schemes in the modern silicon technology. At radio frequency, parasitic inductances or wire-coupling capacitances can be as sensitive as their resistance counterpart or even more sensitive [39]. For example, the sensitive self and mutual inductances cause severe signal ringing and inductive crosstalk between the signal paths. Since most of the interconnecting wires are much smaller in current nanometer processes, the interconnect delay becomes more significant compared to gate delay. Thus, traditional RC parasitic extraction is insufficient to model interconnects in RF layouts. Complete RLC interconnects must be considered for achieving desired layout functionality at RF.

2.3.2 Layout Symmetry and Matching

Device matching and layout symmetry are of the utmost importance to high performance analog and RF circuits [41]. The threshold voltage, mobility and current-factors of MOS transistors are affected by process variation, dopant concentration and gradients in temperature, stress and oxide thickness [23]. These factors may cause a finite mismatch due to asymmetry in layout structures or locations of transistors which are

designed to behave identically. Such mismatch drastically affects analog circuit performance, such as causing DC offsets, finite even-order distortion and lower common-mode rejection. These mismatch effects become worse in modern technologies with smaller device sizes and reduced voltage swing.

In order to minimize the performance sensitivities with respect to layout geometries, some layout structures are required to be matched. Structures of differential devices and symmetric interconnect wires should be placed identically to avoid a mismatch (e.g., $wire\_length1 = wire\_length2$). Matching a pair of devices or wires by a ratio is also required at times (e.g., $structure1 = 0.5 \times structure2$) as discussed in [28] [40]. Moreover, large analog layouts with multiple functional cells may require not only matched transistors, but also matched layout cells [42]. Concretely, both member layout cells for a cell-matching constraint are required to be placed symmetrically to ensure their similar effects regarding processes and temperature gradients. Meanwhile, the matching of passive devices (e.g., on-chip resistors and capacitors) in large analog layouts significantly affects circuit performance as well. These passive devices need to be placed identically to minimize the parasitic effects in analog layout design.

Besides the matching of device and layout cells, interconnect wiring structures are often required to be placed symmetrically with respect to one or several common axes [43]. Matching these layout-induced parasitics is very important to minimize the performance sensitivity to such parasitics. Failure to do so may bring higher offset voltages, larger performance sensitivities and degraded power-supply rejection ratio. For example, certain large performance sensitivity due to two individual parasitics within a differential pair can be reduced to almost zero if both member parasitics are matched.

Given a constraint of matched parasitics, the parasitic-related geometric structures, which are supposed to be identical by design, are always forced to vary similarly in the layout optimization process. Thus, to achieve an effective parasitic-aware layout algorithm, we advocate symmetric-layout designs with enforced matching constraints.

2.4 Summary

In this chapter, a literature review of prior analog design methodologies was presented. Previous attempts for analog/RF layout automation were surveyed. Moreover, the impacts of layout parasitics, symmetry and matching were discussed.

3. Fundamentals of Template-Based Layout Retargeting

Performance of analog/RF integrated circuits is significantly affected by layout parasitics, including both device and interconnects parasitics. The focus of this dissertation is the parasitics of interconnects. The preliminary work of this dissertation is the template-based parasitic-aware layout retargeting/optimization method as reported in [9] (i.e., called PB-RC hereafter). PB-RC is a systematic method of layout generation for technology migration and performance retargeting, which is an effective solution to the parasitic-aware layout retargeting problem. To ensure the desired circuit performance, parasitic bounds are determined from simulations first. These bounds are used to constrain the related layout geometries while retargeting existing high-quality layouts across new technology and new specification sets. The problem is solved using a graph-based algorithm combined with nonlinear programming. Before illustrating the proposed performance-constrained analog/RF retargeting algorithms, the fundamentals of the preliminary work are overviewed in this chapter.

3.1 Template-Based Layout Retargeting

The template-based layout design method was first introduced in [44]. Template-based layout retargeting, which is an optimization process, refers to the generation of a target layout from an existing one [9] by solving a layout template. This process is especially useful when performing layout design for process migration and performance retargeting. Within the retargeting process, a set of constraints corresponding to

technology design rules, layout symmetry, geometry proximity, etc. is first extracted from an existing fine-tuned layout. These constraints are maintained during the optimization process to force the target layout to retain floorplan, symmetry and other properties owned by the original layout while still meeting other updated constraints. The parasitic-exclusive retargeting problem can be formulated as a general layout compaction problem [45] that can be solved separately in the horizontal and vertical directions. The goal of the retargeting is to generate a target layout with the minimum area, while meeting a complete set of constraints (i.e., the updated template). For instance, the layout template for the horizontal direction is illustrated in Figure 3-1 and the template constraints are mathematically represented as:

$$\textit{Minimize} \quad (x_R - x_L), \tag{3.1}$$

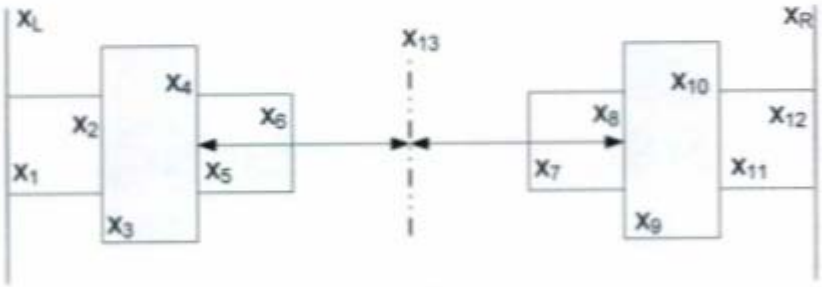
$$\textit{Subject to} \quad x_i - x_j \geq LB, \tag{3.2}$$

$$x_i - x_j = EB, \tag{3.3}$$

$$x_i - x_j = x_k - x_l, \tag{3.4}$$

where x_R and x_L represent the rightmost and leftmost edges of the target layout, and *LB*/*EB* represents related lower-bound/exact-bound. The geometric parameters (e.g., x_i, x_j and x_k) in (3.1)-(3.4) correspond to left/right edges of layout rectangles or a symmetry axis. In Figure 3-1, x_1 to x_{12} are horizontal coordinates of rectangle edges, while x_{13} is the symmetry axis. Eqs. (3.1) - (3.4) are constraints related to design rules, fixed device sizes and layout symmetry. For instance, $x_2 - x_1 \geq 2$ is a minimum-width constraint,

$x_i - x_j = EB$ is a fixed-length constraint, and $x_{13} - x_4 = x_9 - x_{13}$ is a symmetry constraint for a pair of transistors in reference to x_{13}.



(a)

Min $(x_R - x_L)$ subject to

$x_1 - x_L \geq 0$ $x_3 - x_L \geq 0$ $x_2 - x_1 \geq 2$

$x_R - x_{12} \geq 0$ $x_R - x_{10} \geq 0$ $x_{12} - x_{11} \geq 2$

... ...

$x_3 - x_2 = 0$ $x_{11} - x_{10} = 0$

... ...

$x_4 - x_3 = x_{10} - x_9$ $x_{13} - x_4 = x_9 - x_{13}$

... ...

(b)

Figure 3-1 A template formulation in horizontal direction: (a) a two-transistor layout, (b) the extracted template including objective function and linear constraints.

A whole set of extracted constraints forms a symbolic template. This template can be further updated with both automatically generated and user-imposed constraints, such as performance constraints in the proposed formulation of this dissertation, new technology process, new device size, new symmetry requirements, etc. The target layout can be generated by solving the updated template.

3.2 Parasitic-Aware Template-Based Layout Retargeting

Layout parasitics significantly influence circuit performances. Analog/RF circuit performance is actually a function of layout parasitics. To ensure desired circuit performance, parasitic resistances and capacitances must be optimized when retargeting analog circuits. Moreover, parasitic inductances and wire-coupling capacitances must be handled when retargeting RF circuits besides traditional parasitic resistances and capacitances. Thus, parasitic-aware template-based layout retargeting refers to the template-based layout retargeting problem considering parasitic issues.

The parasitic-aware layout retargeting flow for IPRAIL is illustrated in

Figure **3-2**. As shown in

Figure **3-2**, the retargeting system is composed of two modules: the *template extractor* and the *layout generator*. First of all, the layout template extractor identifies the active and passive devices, detects device matching and symmetry, and extracts device connectivity and net-topology from the original layout in *Caltech Intermediate Format* (CIF) [46]. Then users can add additional constraints such as target design rules, new device sizes, new symmetry requirements and parasitic constraints to update the template.

The layout generator solves the complete template and generates a set of geometry solutions. These solutions are mapped back to construct a target layout in CIF. Thus, this flow realizes analog IP reuse by effectively incorporating the embedded expertise in the original layout into the target-layout generation process.

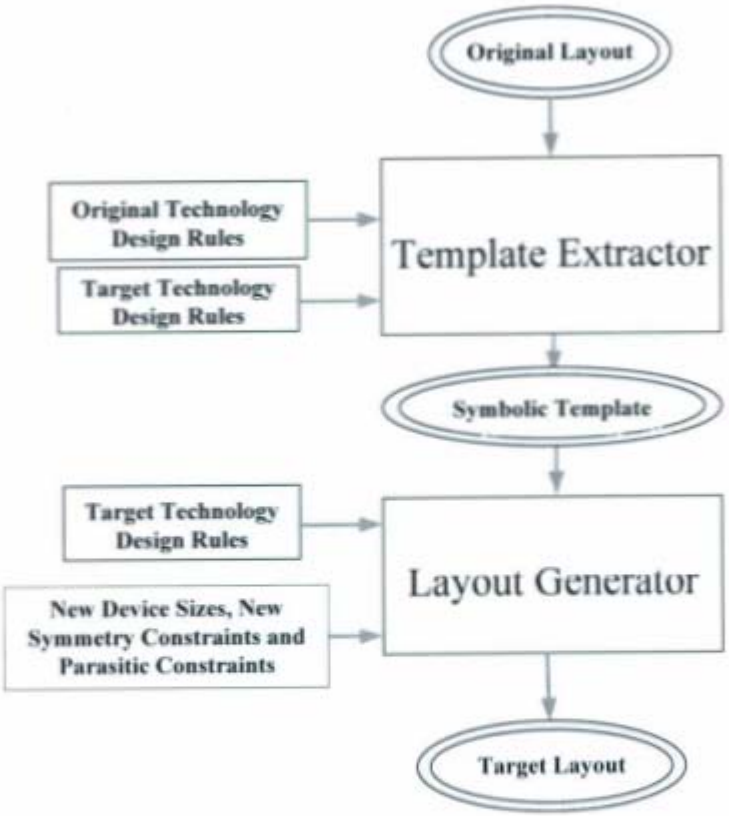


Figure 3-2 A parasitic-aware analog retargeting flow.

3.2.1 Layout Template Extraction

Analog IP reuse is inevitable when retargeting existing analog/RF designs for new technologies or updated performance specification. Macrocell-based placement and routing methodologies have difficulties in incorporating the layout designers' expertise into target layout generation processes, and thus usually generate worse target layouts compared to handcrafted layouts. To overcome this disadvantage, template-based layout automation has been proposed in [4]. This methodology extracts a symbolic template including floorplan, symmetry, and device/wiring alignment information from the existing high-quality layout. Mathematically, the symbolic template is a set of constraints on the target layout geometry. User-imposed constraints can be incorporated into the extracted template to form a complete template which can be solved to generate a target layout.

3.2.2 Parasitic Constraints

Device parasitic models are simply extracted once the device identification is completed following the device identification method advocated in [47]. For the extraction of interconnect parasitics, the current flow direction must be determined for resistance calculation [9]. The algorithm in [48] is adopted in our approach for this purpose. Once parasitics are extracted, sensitive nets are identified and marked with their resistance and capacitance upper bounds as well as available matching requirements. For multi-finger transistors (either gate-connected or diffusion-connected), resistance and capacitance values are approximated and subtracted from their upper bounds.

To obtain successful analog circuit performance, parasitic resistances and capacitances must be restricted within their upper bounds. Thus, the formulation of parasitic-aware retargeting is a general layout compaction problem along with some geometric constraints due to parasitic bounds. It is worth mentioning that the parasitic expressions of resistance and capacitance are nonlinear functions of layout geometries involving both horizontal and vertical dimensions. The geometric constraints due to parasitics have the general form of:

$$P(x, y) \leq upperBound \quad , \tag{3.5}$$

where *upperBound* is a constant, and x, y are layout coordinates in horizontal and vertical dimensions. These bounding constraints are represented with (3.6) and (3.7) for parasitic resistance and wire-substrate capacitance, respectively. Moreover, parasitic matching constraints must be enforced to avoid a performance loss as shown in (3.8) and (3.9).

$$\sum \rho_{sh} \frac{len}{wid} + R_{MFT} + R_{cont} \leq UB_{RES} \ , \tag{3.6}$$

$$\sum c_a \times len \times wid + \sum c_{sw} \times 2 \times len + C_{MFT} \leq UB_{CAP} \ , \tag{3.7}$$

$$\left[\sum \rho_{sh} \frac{len}{wid} + R_{MFT} + R_{cont} \right]_{Structure\ 1} = \left[\sum \rho_{sh} \frac{len}{wid} + R_{MFT} + R_{cont} \right]_{Structure\ 2} , \tag{3.8}$$

$$\begin{aligned} &\left[\sum c_a \times len \times wid + \sum c_{sw} \times 2 \times len + C_{MFT} \right]_{Structure\ 1} = \\ &\left[\sum c_a \times len \times wid + \sum c_{sw} \times 2 \times len + C_{MFT} \right]_{Structure\ 2} \end{aligned} , \tag{3.9}$$

where UB_{RES} and UB_{CAP} are the upper bounds of parasitic resistance and wire-coupling capacitance for sensitive nets, respectively, *len* and *wid* represent the length and width of a layout tile, respectively, ρ_{sh} is the sheet resistance per unit length, c_a is a substrate capacitance per unit area, c_{sw} is wire-substrate capacitance per unit length, R_{MFT} refers to the parasitic resistance for multi-finger transistors, R_{cont} refers to contact resistance, and C_{MFT} refers to parasitic capacitance for multi-finger transistors.

The nonlinear bounding constraints as shown in (3.6) – (3.9) are solved by nonlinear programming to generate a set of constants as the right-hand-side of related linear constraints. These linear constraints are shown as:

$$x_i - x_j \geq const, \quad y_i - y_j \geq const, \tag{3.10}$$

$$x_i - x_j \leq const, \quad y_i - y_j \leq const, \tag{3.11}$$

$$x_i - x_j = x_k - x_l, \quad y_i - y_j = y_k - y_l, \tag{3.12}$$

where x_i and x_j are layout geometric parameters in horizontal direction, y_i and y_j are layout geometric parameters in vertical direction, and x_k and y_k are geometric parameters representing symmetry axis in horizontal and vertical directions, respectively, and *const* refers to a bound. (3.10) and (3.11) refer to linear constraints due to lower and upper-bound constraints while (3.12) represents the linear constraints due to matching parasitics.

In PB-RC, the parasitic bounds used in right-hand-side of (3.6) and (3.7) are generated by manually or semiautomatic simulations which are not directly related to layout geometries. These predicted bounds are not accurate and may bring over-

constraints to the formulation. Moreover, the correlation among all parasitics is not considered in a global way.

3.2.3 Layout Compaction

The formulation of layout solving problem is nonlinear due to the nonlinearity and two-dimension features of parasitic bounding constraints as shown in (3.6)-(3.9). Figure 3-3 shows the two-dimensional layout generation flow. The integration of new design rules and device sizes, and the post-processing of the retargeted layout using the longest-path algorithm are conducted simply following the techniques shown in [4]. After the symbolic template is updated by nonlinear parasitic constraints as well as symmetry constraints, a nonlinear problem is formulated.

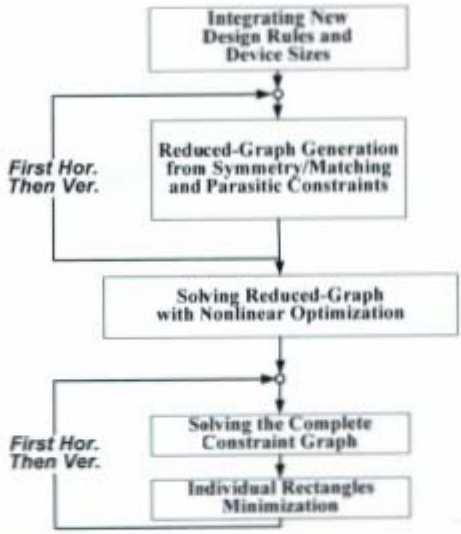


Figure 3-3 Two-dimensional layout generation flow.

To improve the computational speed, graph compaction techniques are adopted to convert linear equations or inequalities into a graph form. An original graph is formed by converting all linear constraints in the updated layout template (e.g., minimum width constraints).

However, since nonlinear expressions or 4-parameter equations are not allowed in graph techniques, these constraints like parasitic constraints as shown in (3.6) - (3.7) or symmetry constraints as shown in (3.12) are kept as equations/inequalities which are handled later in the reduced-graph solving phase. In Figure 3-3, the layout solving includes three steps: (1) generation of the reduced graph from symmetry and parasitic constraints; (2) solving the reduced graphs with nonlinear programming; (3) solving the complete constraint graph with longest-path algorithm. Here, a reduced graph is a simplified graph which is equivalent to the original constraint graph but with much fewer nodes and arcs. To ensure the equivalence between the original graph and the reduced one, weights of arcs between nodes are calculated by applying any longest-path algorithm (e.g., Bellman-Ford algorithm in this work) from each core node to the other core nodes in the reduced graph. A set of new arc weights is then obtained and the updated core graph is converted to a set of equations and/or inequalities. By combining both horizontal and vertical converted linear constraints with the nonlinear parasitic constraints and 4-parameter symmetry/matching constraints, the complete reduced graph is then constructed. This constraint graph can be solved by a nonlinear solver (e.g., IPOPT [49] in this dissertation).

The complete core graph may contain thousands of linear constraints and a very small number of nonlinear parasitic constraints for medium or large sized designs. It is

slow to handle all these constraints (e.g., 95% linear constraints mixed with 5% nonlinear constraints) simultaneously. To improve the searching efficiency of the nonlinear solver, a two-phase solving scheme is designed. In the first phase, a nonlinear-only compaction (i.e., considering only parasitic and symmetry/matching constraints) is solved using graph-based optimization, combined with nonlinear programming. In the second phase, a linear-only compaction is conducted with graph-based optimization and a longest-path algorithm. The solutions out of the first phase act as the start points of the second phase to separate the searching effort from a complete LP (i.e., linear programming) to a complete NLP (i.e., nonlinear programming). This scheme greatly improves the efficiency of the parasitic-aware layout solving.

A set of optimal solutions is found by the nonlinear solver as a set of new arc weights for symmetry/parasitic nodes. By incorporating these optimal arc weights into the original graph, a complete graph, which includes all required constraints in an LP form, is formed. This complete graph can be finally solved to generate a set of solution geometries that forms a target layout.

3.3 Experimental Environment

The coding work for the proposed algorithms (as presented in Chapters 4, 5, and 6) was conducted in *C/C++* under *UNIX* system on a *Sun Blade 100* server.

Circuit netlists and simulation scripts were obtained from *Cadence* [50] *Analog Design Environment*. The simulators involve HSPICE [10], SPECTRE [51] and SPICE3 [52] for analog circuits. For the double-ended LNA as shown in Figure 6-11, we took the

advantage of *Cadence Ocean Script* [53] to run multiple simulations across different sweep variables in a limited number of iterations. Moreover, HSPICERF [11] was used as the simulator for the NMOS LNA as shown in Figure 4-6.

The solvers of MOSEK [54] and LINDO [55] were involved as the linear programming solvers for the experiments in Chapter 4. Moreover, IPOPT [49] was used as the nonlinear and MINLP solver for the experiments in Chapter 5 and Chapter 6.

3.4 Summary

In this chapter, the preliminary template-based layout retargeting algorithm was presented. The fundamental flow of template extraction and layout compaction were illustrated. The parasitic bounding constraints were discussed. The implementation environment for the research in this dissertation was laid out. In the following chapters, the improved layout design algorithms for analog and RF layout retargeting will be presented.

4. Generation of Parasitic Solutions for Analog and RF Circuits

This chapter introduces an algorithm for generating parasitic solutions with the aid of performance sensitivities. The proposed flow is composed of three steps: parasitic bound generation, sensitivity computation, and parasitic solution generation [8]. A bisearch algorithm is developed to automatically generate parasitic upper bounds from simulations. Sensitivity computation techniques are analyzed. The parasitic-constraint generation within this chapter regards parasitic parameters as design variables, without considering layout geometries related to these parasitics. The proposed flow has been implemented in a *C++* package that automatically calls circuit simulators and linear programming solvers. The experimental results exhibit its effectiveness and efficiency in generating parasitic solutions for several analog/RF circuits.

4.1 Introduction

The previous attempts to generate parasitic solutions using performance sensitivities were introduced in [56] [57] [58]. This chapter introduces an improved algorithm that automatically generates parasitic solutions for analog/RF circuits. The performance of analog and RF integrated circuits is very sensitive to physical issues such as device style and interconnect parasitics. The parasitics, which include resistances and capacitances, must be optimized by enforcing some bounding constraints or parasitic-induced performance-deviation thresholds. In the proposed formulation for this chapter, the

parasitic bounds are automatically generated by using a bisearch algorithm that calls a series of circuit simulations. With the aid of sensitivity techniques, performance sensitivity is calculated to represent the parasitic impacts on circuit performances. The performance deviation is then modeled as the product of parasitic parameters and their performance sensitivities. This deviation is restricted within certain thresholds, in the proposed performance constraints, to ensure a success of circuit performance.

The proposed parasitic-constraint generation algorithm has the following features:

(1) Upper bounds of unmatched and matched parasitics are automatically generated to offer linear-programming (LP) solvers maximum optimization flexibility;

(2) Since performance sensitivities may vary along with changing parasitics, central-difference performance sensitivities are applied;

(3) Moreover, a performance-constrained formulation, which can be readily solved by any standard LP solver, is constructed.

The rest of this chapter is organized as follows. Section 4.2 presents the problem definitions. Section 4.3 introduces a bisearch simulation-based parasitic bound generation algorithm. The formulation of the proposed parasitic problem and the design flow are shown in Section 4.4. Experimental results are reported in Section 4.5. Section 4.6 gives a brief chapter summary. In order to better illustrate the later sections, the notations used in this chapter are defined in Table 4-1.

Table 4-1 Notations list for Chapter 4.

Symbol	Description
W_i	*A circuit performance parameter*
P_j	*A parasitic parameter*
S_{ij}	*The sensitivity of W_i with respect to P_j*
ΔW_i	*The performance deviation of W_i due to parasitics*
W_{i-spec}	*The performance specification for W_i*
ΔW_{i-max}	*Maximum allowed deviation for W_i*
$P_{j-solution}$	*A parasitic solution of P_j*
$P_{j-ubound}$	*An upper bound of P_j*
P_{j-min}	*A feasible lower limit of P_j*
P_{j-max}	*A feasible upper limit of P_j*

4.2 Problem Definitions

4.2.1 Parasitic Solution Generation

Parasitic solution generation (i.e., generation of parasitic constraints) refers to the generation of a set of optimal parasitic solutions towards desired circuit performance. These solutions may involve parasitic resistances, capacitances and inductances. These solutions are in the form of (4.1) for solutions of non-matching parasitics, and (4.2) for the solutions of matching parasitics.

$$P_k = P_{k-solution} \quad , \tag{4.1}$$

$$P_{m1} = P_{m2} = P_{m-solution} \quad , \tag{4.2}$$

where P_{m1} and P_{m2} are two member parasitics for a matching pair, and P_k is a non-matching individual parasitic.

For an individual parasitic as shown in (4.1), no matching requirements are enforced, thus only its parasitic value affects circuit performance. In such a case, this parasitic must be limited within a certain upper bound. It is worth mentioning that, some parasitics may improve certain circuit performances. However, a layout tile physically has three parasitics including its resistance, capacitance and inductance, and these three parasitics may have different impacts on same circuit performance (e.g., its resistance improves the AC *gain* but its capacitance counterpart degrades the AC *gain*). Thus, all individual parasitics must be optimized globally to minimize the induced performance degradation.

Parasitic matching is a significant issue in circuit optimization, especially if circuits have differential pairs. Parasitic mismatch can severely degrade circuit performance or even lead to a performance loss for analog/RF products. Since some devices or interconnects are required to function identically, matching parasitics within a matching pair share the same parasitic parameter as shown in (4.2). Moreover, the shared value of a matching pair must also be restricted within an upper bound. Thus, matching parasitics are handled by generating a set of shared solutions for matching pairs.

4.2.2 Performance Sensitivity

Performance sensitivity is used to quantify the dependence of circuit performance with respect to parasitics. The sensitivity of performance W_i with respect to parasitic P_j can be mathematically represented as

$$S_{ij} = \partial W_i / \partial P_j . \tag{4.3}$$

The sensitivity technique, which applies performance sensitivities on net parasitics for analog routing problems, was initially advocated in [56]. In that work, SPICE3 simulations are first conducted to generate performance gradients with respect to circuit parameters (e.g., voltage and current). Performance sensitivities with respect to parasitics are then modeled as functions of circuit-parameter gradients and certain independent variables (e.g., frequency and time). However, that sensitivity calculation method is strongly dependent on the expertise of using SPICE3 and has high computation complexity.

To obtain fast evaluation of performance sensitivities using any circuit simulators, we deploy direct calculation of sensitivity gradients in the proposed algorithm. This scheme takes advantage of readable simulated performances. The parasitics in the original layout are extracted and a circuit netlist including variable parasitics is set up for simulation. Multiple performances can be obtained by varying parasitics in a number of simulations. Let $W_{ij}(P)$ represent the simulated performance of W_i when a parasitic of P_j has a value P ($P \in (LB, UB)$), the sensitivity of W_i with respect to P_j can be calculated using finite-difference approximation as:

$$S_{ij} = [W_{ij}(LB) - W_{ij}(UB)]/(LB - UB), \tag{4.4}$$

where LB/UB refer to the lower/upper bound of the parasitic P_j. The sensitivity calculated with (4.4) approximately reflects the dependence of W_i on P_j within the domain (LB, UB). However, this approximation loses accuracy when the domain (LB, UB) is large since performance sensitivities can vary with changing parasitics. To obtain accurate

performance sensitivities around a parasitic value, we advance the sensitivity computation to a central-difference approximation as:

$$S_{ij} = [W_{ij}(P + \Delta) - W_{ij}(P - \Delta)] / 2\Delta, \tag{4.5}$$

where Δ is a minimum increment (e.g., $\Delta \leq 0.04 \times UB$ in the thesis research). This expression is accurate enough when calculating performance sensitivities around a parasitic value. Therefore, (4.5) is adopted in the proposed algorithm. Here, upper bounds are determined as the values where sensitivities are calculated because our objective is to optimize parasitic parameters towards feasible large values around their upper bounds.

4.2.3 Performance Constraints

Circuit performance degradation due to parasitics can be represented as a function of layout parasitic parameters and related performance sensitivities. This function must be restricted within the maximum allowed performance deviation. To simplify the implementation without too much compromising the accuracy, linear approximation is adopted to expand the total performance deviation into a linear summation of performance deviation due to each parasitic. These performance constraints without considering layout geometries have a general form of (4.6),

$$\sum_j \Delta W_{ij} \leq \Delta W_{\max}, \tag{4.6}$$

where ΔW_{ij} represents the performance deviation due to a certain parasitic, and $\Delta W_{\max}$ refers to the maximum allowed performance deviation due to all parasitics. With

performance sensitivities, a certain performance deviation ΔW_{ij} due to a parasitic P_j can then be expanded into the product of a performance sensitivity and a parasitic parameter as:

$$\Delta W_{ij} = S_{ij} \times P_j. \tag{4.7}$$

Linear approximation is adopted to expand the total performance deviation into a linear summation of performance deviation due to each parasitic. The expanded performance constraints can be represented as:

$$\sum_{j=1}^{N_P} S_{ij} \times P_j \leq \Delta W_{i\max} \quad , \tag{4.8}$$

where N_P represents the number of parasitics affecting the performance covered in (4.8).

4.3 Parasitic-Bound Generation

A bisearch algorithm is designed to automatically generate a set of parasitic upper bounds as shown in the pseudo-code of Figure 4-1. Two pointers called *left* and *right* (lines 2-3 of (a)) are defined. Simulations are always called to search a possible upper bound in the middle of the domain (*left, right*). Concretely, an iteration of simulation is conducted after assigning a parasitic with a pointer named *current* which is the average of *left* and *right* (lines 2-3 of (b)). An initial iteration of simulation is first conducted (lines 2-3 in (a)) to generate a set of starting performances. Based on starting performances, decisions of more simulations are made (lines 4-24 of (a)) to further shrink the searching domain of (*left, right*). The upper bound is finally detected when performance approximately reaches its specification (i.e., simulated performance ≈ its specification).

```
boundGeneration
Begin
 initialization {upperbound, left, right} <= {0, 0, initial};
    simulate (left, right);
    While (upperbound = 0) Do
      if (all performances meet specifications && current = initial)
            upperbound <= initial;
      else if (more than one performance fails to meet specification)
            right <= current;
            simulate (left, right);
      else if (only one performance called W_s fails to meet specification)
            While (W_s != its specification) Do
                  if (W_s fails its specification)
                        right <= current;
                        simulate (left, right);
                  else if (W_s is better than its specification)
                        left <= current;
                        simulate (left, right);
                  else
                        upperbound <= current;
                  end if
      else
            left <= current;
            simulate (left, right);
      end if
End
```

(a)

```
simulate
Begin
      current <= (left+right)/2;
      simulate the netlist given P_f = current;
End
```

(b)

Figure 4-1 A bisearch algorithm for parasitic upper bound generation.

as a set of performance specifications. This flow is simulation-based, which takes advantage of the readable performances from circuit simulations using any simulator of the designer's choice.

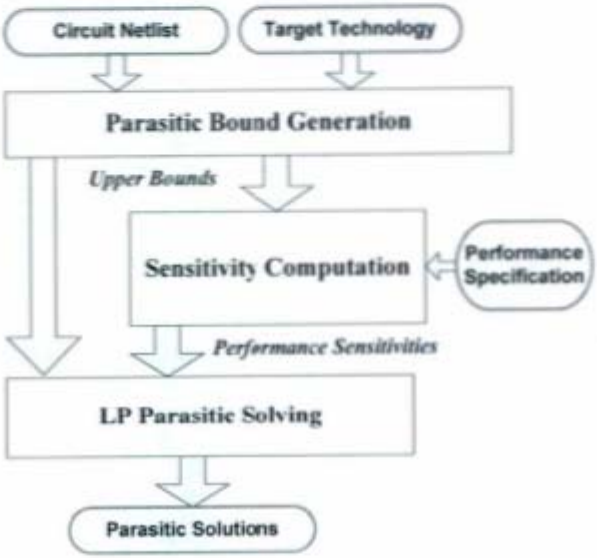


Figure 4-2 Design flow of parasitic-solution generation.

With the circuit netlist incorporating variable parasitics, a bound-search engine is activated to conduct a set of simulations using any standard circuit simulator. For different scenarios defined in the bisearch algorithm as shown in Figure 4-1, a set of simulated performances are setup. These simulated performances as well as the performance specifications are incorperated into a sensitivity engine for sensitivity

computation. Central-difference performance sensitivities with respect to parasitics are then generated by using the method as discussed in Section 4.2.2. These sensitivities are used to construct a set of performance constraints while the generated upper bounds are used to construct a set of bounding constraints in the formulation. The formulated problem is finally solved by an LP solver and a set of parasitic solutions are eventually obtained.

4.4.2 Parasitic-Problem Formulation

The formulation of the parasitic-solution problem is shown in (4.9) - (4.12). The objective of the optimization is to maximize parasitics as shown in (4.9) while meeting performance constraints. It is to be noted that, larger parasitics usually correspond to less complicated layout structures, which in turn provides layout solving with wider flexibility. It can be seen that the formulation above is a linear programming problem. Due to the linearity of its objective function and constraints, the formulated problem can be readily modeled and solved with a standard LP solver.

The objective function is modeled as linear summation of all parasitic parameters limited by their weights as shown in (4.9). The reciprocal of the upper bound ($P_{j\text{-}ubound}$) is used to quantify the contribution of P_j to the overall objective function. These reciprocals are designed to ensure that a relatively large parasitic solution is generated if its upper bound is relatively large. For matching parasitics, a single parameter is used to represent the identical value for a pair of matched parasitics in the performance constraints of (4.10).

$$Maximize \quad \sum_{j=1}^{N_P} (1/P_{j\text{-bound}})P_j \tag{4.9}$$

Subject to:

$$\sum_{j=1}^{N_P} S_{ij} \times P_j \le \Delta W_{i\max} \tag{4.10}$$

$$P_{j\text{-min}} \le P_j \le P_{j\text{-max}} \tag{4.11}$$

$$P_j \le P_{j\text{-ubound}} \tag{4.12}$$

A set of parasitic bounds $\{P_{j\text{-ubound}}\}$ can be generated to provide the LP-solver with maximum solving flexibility as shown in (4.12). However, our desired flexibility must simultaneously consider both solving feasibility and solution usability. To make solutions usable, minimum and maximum feasible parasitic values ($P_{j\text{-min}}$ and $P_{j\text{-max}}$ in (4.11)) for a layout are enforced in (4.11). Extremely large or small parasitic values are meaningless to a layout due to the following reasons: First, completely eliminating parasitic effect (i.e., $\{P_j\}=\{0\}$) is unreachable for a real layout; second, layout parasitics can only reach finite large values due to finite layout geometries (i.e., $P_{j\text{-ubound}} \le P_{j\text{-max}}$ always holds).

4.5 Experiments

The proposed parasitic-solution generation flow (i.e., called P-LP hereafter) has been implemented in *C++* language under *UNIX* system. The commercial solver MOSEK is used as our LP solving engine. In this section, we present the results of sensitivity-based parasitic-solution generation on several analog/RF circuits: a *two-stage Miller-*

compensated operational amplifier (called opamp hereafter) as shown in Figure 4-4, a *single-ended folded-cascode opamp* as shown in Figure 4-5 and an *NMOS cascode LNA* as shown in Figure 4-6. The three circuits are designed in *TSMC18* technology.

To demonstrate the superior effectiveness of the proposed approach (i.e., P-LP), an alternative approach (called P-UC hereafter) is setup using UC-Berkeley sensitivity computation techniques in [56] following the flow introduced in [57]. P-UC approach calculates performance sensitivities using voltage/current sensitivities and parameters of frequency/time with the aid of a specified simulator SPICE3. In contrast, P-LP features direct sensitivity computation with reasonable accuracy by simulating variable parasitics using any simulators. An RC π-model [59] as shown in is used to represent interconnect parasitics in the circuit netlists for analog designs. The nets of sensitive parasitics are indicated in the schematics. HSPICE is used as the simulator for both opamps and HSPICERF for the LNA.

HSPICE/HSPICERF is one of the most powerful analog/RF simulators. These simulators feature effective and fast evaluation of readable Transient, DC and AC performances.

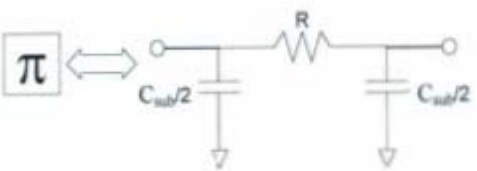


Figure 4-3 An RC π-model for a parasitic interconnect.

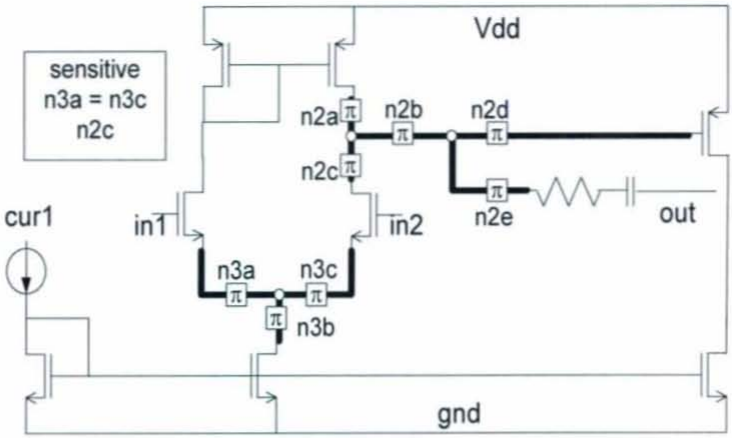


Figure 4-4 A two-stage Miller-compensated opamp.

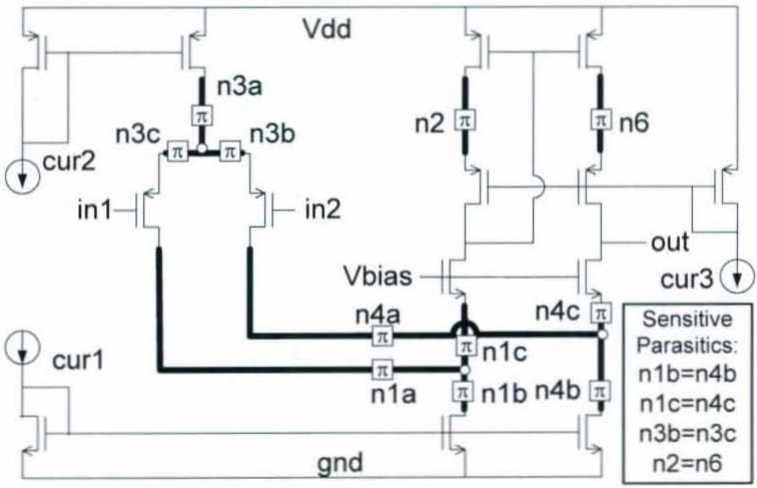


Figure 4-5 A single-ended folded-cascode opamp.

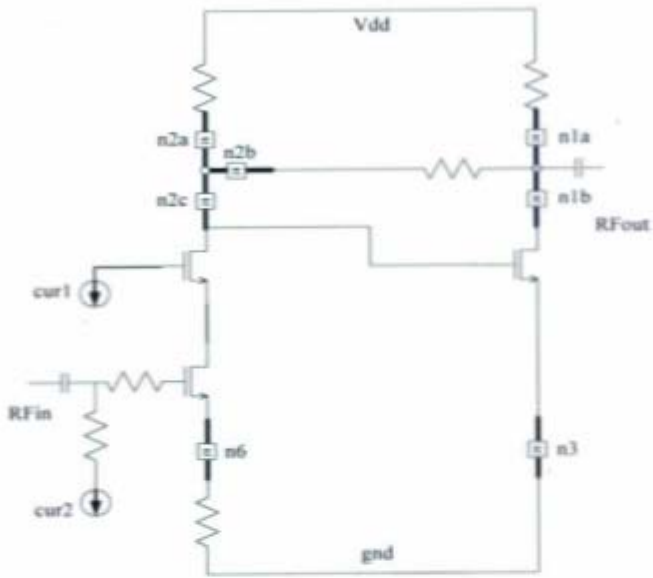


Figure 4-6 An NMOS LNA.

First of all, parasitic solutions were generated by P-LP and P-UC on the operational amplifiers. With the same parasitic upper bounds, sensitivity computation was conducted using the aforementioned central-difference scheme (within P-LP) compared to the traditional sensitivity techniques within P-UC. The comparisons of some generated performance sensitivities and simulated circuit performances are reported in Table 4-2. To simplify the comparison, only the performance of AC gain is considered and sensitivities of the critical parasitics are reported. The observations are: (1) sensitivities generated by both methods are closely correlated (i.e., almost proportionally); (2) P-UC sensitivities are generally much smaller than P-LP ones, meaning a worse case of

sensitivity is generated using P-LP that tightly constraints the performance deviation; (3) AC gain achieved by P-LP solutions meets the specification (i.e., *gain* $\geq$ 60) for both circuits whereas P-UC produces a performance loss (*gain* = 57.5) for the cascode opamp. This performance loss may result from the setting of a special threshold which was used in [58]. In that paper, there is no clue how to set a proper threshold for different applications. Therefore, the direct sensitivity scheme was adopted in P-LP.

Table 4-2 Sensitivities and simulated performances for P-LP/P-UC.

	Parasitic Res. (Ω)	Sensitivities of Gain (dB/ Ω)		Solution Performance	
		P-LP	P-UC	P-LP	P-UC
Two stage opamp	***R3a***	-0.0324	-0.0020	Gain= 64.0	Gain= 60.1
	R3c	Match	Match		
	R2c	-0.1450	-0.0296		
Folded-cascode opamp	***R3a***	-0.0046	-0.0007	Gain= 60.6	Gain= 57.5
	R3b	-0.0096	-0.0012		
	R3c	Match	Match		
	R2	-0.0068	-0.0021		
	R6	Match	Match		

Then, the P-LP algorithm was used to conduct parasitic-solution generation on the aforementioned three circuits. Performances of AC gain, bandwidth and phase margin are considered for both opamps. RF performances of S-parameters and noise figure are considered for the LNA. Performance specifications, nominal performance values and maximum allowable performance deviations are listed in Table 4-3. For instance, the AC

gain specification of the two-stage opamp is 60 dB, its nominal gain (without parasitic impacts) is 64.5 dB, and its maximum allowed deviation is -4.5 dB.

Table 4-3 Performance Requirements.

		Gain (dB)	**Bandwidth (MHz)**	**Phase Margin (°)**
Two-stage Opamp	W_{i-spec}	60.0	100.0	90.0
	ΔW_{i-nom}	64.5103	107.3	90.5290
	ΔW_{i-max}	-4.5103	-7.3	-0.5290
Cascode Opamp	W_{i-spec}	60.0	60.0	60.0
	ΔW_{i-nom}	60.6473	63.44	61.1924
	ΔW_{i-max}	-0.6473	-3.44	-1.1924
Cascode LNA		**S12(dB)**	**S21(dB)**	**Noise Figure(dB)**
	W_{i-spec}	-25	-30	30
	ΔW_{i-nom}	-36.2053	-18.7896	26.7383
	ΔW_{i-max}	11.2053	-11.2104	3.2617

The parasitic upper bounds, performance sensitivities and generated parasitic solutions for parasitic resistances and capacitances are reported in Table 4-4 for the two-stage opamp, Table 4-5 for the cascode opamp and Table 4-6 for the LNA. For instance, in Table 4-4, the matching parasitic of *R3a=R3c* corresponds to the shared parasitic resistance in the differential pair, which has an upper bound of 16.2 Ω. The sensitivities of AC gain, bandwidth and phase margin with respect to this parasitic are -0.032 dB/Ω, -0.48 MHZ/Ω and -0.0086 °/Ω, respectively. The generated parasitic solution for this parasitic is 16.2. Moreover, in Table 4-6, the individual parasitic *R1* has an upper bound

of 50 Ω. The sensitivities of *S12*, *S21* and *noise figure* with respect to this parasitic are 0.0025 dB/Ω, -0.0013 dB/Ω, and 0.0002 dB/Ω, respectively. The generated solution of this parasitic is 50 Ω. It is worth observing that, some parasitics can improve certain circuit performance since their reported performance sensitivities are positive. For example, the bandwidth sensitivity for *R10* is 0.4 MHZ/Ω as shown in Table 6. Performance sensitivities of wire-substrate capacitances are also included in the tables although they are far less sensitive than their resistance counterparts.

Table 4-4 Upper bounds, performance sensitivities and solutions of parasitic resistances and capacitances for the two-stage opamp.

Parasitic	Bounds (Ω, fF)	Sens. (Gain)	Sens. (Bandwidth)	Sens. (PhaseMargin)	Solutions
R3a=R3c	16.2	-0.032	-0.48	-0.0086	16.2
R3b	78.9	-0.057	-0.08	-0.0014	78.9
R2a	48.5	0.031	0.24	-0.0471	48.5
R2b	100	-0.001	-0.007	-0.0043	100
R2c	35.1	-0.145	-0.11	-0.0023	25.0
R2d	47.9	0.0001	0	-0.0113	45.9
R2e	100	0.0012	0.4	0.13	100
R10	100	0	0.4	0.1299	100
C3a=C3c	1e-12	0	-0.01	0.0034	1e-12
C3b	1e-12	0	0	0.0014	1e-12
C2a	9e-14	0	-0.023	-0.0592	8.8e-14
C2b	9e-14	0	-0.023	-0.0592	8.8e-14
C2c	9e-14	0	-0.023	-0.0592	8.8e-14
C2d	9e-14	0	-0.023	-0.0592	8.8e-14
C2e	9e-14	0	-0.023	-0.0592	8.8e-14
C10	1e-12	0	0	0.0121	5.6e-13

Table 4-5 Upper bounds, performance sensitivities and solutions of parasitic resistances and capacitances for the cascode opamp.

Parasitic	Bounds (Ω, fF)	Sens. (Gain)	Sens. (Bandwidth)	Sens. (Phase Margin)	Solutions
R1b=R4b	200	-0.0029	0	0.0003	200
R1c=R4c	200	0.0013	0	-0.0015	200
R3b=R3c	146	-0.0096	-0.02	0.0056	3.6
R2=R6	117	-0.0136	0	-0.0007	3.9
R1a	85.9	-0.0036	0	-0.0002	85.9
R4a	200	0.0031	0	-0.0005	200
R3a	200	-0.0046	0	0	200
R5a	200	0.002	0	-0.0002	200
R5b	200	-0.0114	0	0.0017	5.1
R5c	200	-0.0085	0	-0.0008	81
R5d	200	-0.0278	0	0.001	5.7
R5e	200	-0.0175	0	0.0008	6
R8a	200	-0.0023	0	-0.0001	200
R8b	200	0.0072	0	-0.0004	200
C1b=C4b	27.6	0	-0.01	-0.0426	27.6
C1c=C4c	27.6	0	-0.01	-0.0426	27.6
C3b=C3c	17.1	0	0.01	-0.0076	17.1
C2=C6	200	0	-0.02	0.0014	118.9
C1a	56.7	0	-0.01	-0.0202	56.7
C4a	56.7	0	0	-0.0207	56.7
C3a	34.2	0	0	-0.0038	34.2
C5a	38.3	0	-0.02	-0.0298	38.3
C5b	38.3	0	-0.02	-0.0298	38.3
C5c	38.3	0	-0.02	-0.0298	38.3
C5d	38.3	0	-0.02	-0.0298	38.3
C5e	38.3	0	-0.02	-0.0298	38.3
C8a	41.6	0	-0.08	0.0356	41.6
C8b	41.6	0	-0.08	0.0356	41.6

Table 4-6 Upper bounds, performance sensitivities and solutions of parasitic resistances and capacitances for the NMOS LNA.

Parasitic	Bounds (Ω, fF)	Sens. (S12)	Sens. (S21)	Sens. (Noise Figure)	Solutions
R1	50	0.0025	-0.0013	0.0002	50
R2	50	-0.0126	0.0070	-0.0024	50
R3	50	-0.0189	0.0026	-0.0021	50
R4	50	0.0570	-0.0201	0.0041	50
R5	17.3	-0.1591	-0.1869	0.0540	13.7
R6	50	0	0	0	50
R7	38.5	0.0366	-0.1272	0.0203	21.2
C1	50	-0.0023	-0.0023	0.0006	50
C2	50	-0.0023	-0.0023	0.0006	50
C3	50	-0.0023	-0.0023	0.0006	50
C4	50	0	0	0	50
C5	50	0.0141	0.0123	-0.0047	50
C6	50	-0.0060	-0.0060	0	50
C7	50	-0.0060	-0.0060	0	50

Table 4-7 Time efficiency statistics for P-LP.

Circuits	Time (s)
Two Stage OPAMP	397
Cascode OPAMP	553
NMOS cascode LNA	116

4.6 Summary

In this chapter, an algorithm of automatic parasitic-solution generation was presented for analog/RF circuits. An automatic parasitic-bound generator was designed. Sensitivity computation techniques and performance constraint models were demonstrated. The proposed flow was implemented in *C++* language and experiments of three analog and RF circuits were conducted. The efficiency and effectiveness of our proposed approach were reported.

Although the parasitic upper bounds and performance sensitivities generated by this algorithm can be further used in the layout generation problems, the algorithm proposed in this chapter was limited to a simulation-based parasitic control. The algorithm does not physically relate parasitic parameters to their layout geometries. To achieve a geometry-involved layout optimization flow, parasitic parameters will be expanded into geometric structures in Chapters 5 and 6 to directly link the performance constraints with layout geometries for analog/RF designs.

5. Performance-Constrained Parasitic-Aware Retargeting of Analog Layouts

This chapter presents a template-based algorithm that conducts automatic performance-constrained parasitic-aware retargeting of analog layouts. As a further extension of Chapter 4, which only considers variable-based parasitics (i.e., no geometries involved), layout geometries are incorporated into the formulation of analog layout retargeting. The algorithm applies a piecewise-sensitivity model to limit parasitic-related layout geometries by directly constructing performance constraints subject to maximum allowed performance deviation [6]. The formulated problem is finally solved using graph-based techniques combined with mixed-integer nonlinear programming. The algorithm has been incorporated into IPRAIL [7]. It is demonstrated to be effective and efficient when retargeting analog layouts to new technologies and/or updated specifications.

5.1 Introduction

The preliminary work in [9] presents a parasitic-aware optimization flow as an effective solution to analog layout retargeting. In that work, a set of parasitic bounds are manually estimated through simulations before layout generation. These bounds are used to constrain corresponding geometric parasitic expressions. However, the parasitic bounds are obtained from circuit simulations without considering layout geometries.

Moreover, each parasitic bound is enforced in certain separate parasitic constraints. This scheme fails to consider the correlation and/or cancellation among all the parasitics that can affect the same performance. Thus, the control of parasitics is not handled in a global manner. More important, the bounding limits may over-constrain the problem or even render it infeasible.

To achieve a global performance optimization with accurate parasitic control, this chapter presents a performance-constrained sensitivity-based retargeting algorithm. This approach features direct performance constraints in a geometry-based global optimization. Accurate performance sensitivities due to parasitics are applied by adopting a piecewise-sensitivity model. The parasitic-aware performance-constrained retargeting is formulated as a mixed-integer nonlinear programming problem. Compared to the bound-based retargeting method [9] (PB hereafter within this chapter), better solvability is achieved by removing the over-constraints due to the imposed parasitic bounds.

The rest of the chapter is organized as follows. Section 5.2 introduces the proposed design flow. Section 5.3 discusses the mathematical modeling scheme in our approach. We introduce the piecewise sensitivities in Section 5.4. The parasitic-problem formulation is provided in Section 5.5. Section 5.6 reports the experimental results followed by a chapter summary made in Section 5.7.

5.2 Design Flow

Figure 5-1 shows the proposed performance-constrained layout retargeting flow. This flow has three major steps including the sensitivity computation, the layout template

extraction and the layout generation. This flow handles process migration as well as performance retargeting for analog layouts with the capability of incorporating user-imposed constraints (e.g., new symmetry requirements) into optimization.

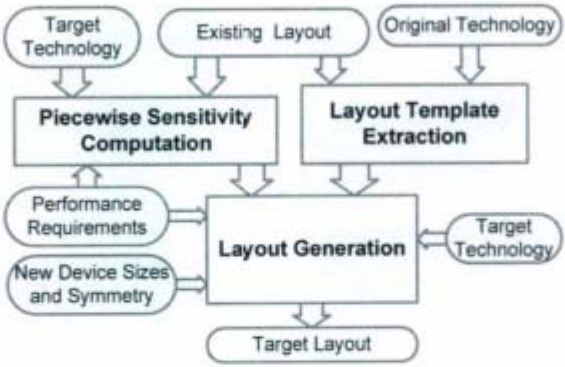


Figure 5-1 A performance-constrained analog layout retargeting flow.

After a netlist is extracted from an original layout, target technology and performance requirements are incorporated to form an updated netlist. A sensitivity engine then automatically conducts a series of simulations to generate piecewise performance sensitivities with respect to variable parasitics in the updated netlist. The generated piecewise sensitivities are used to construct MINLP performance constraints. Given the original technology design rules, a layout template extractor automatically extracts all the expertise embedded in an existing layout to construct a symbolic template, which is composed of a set of linear constraints. This template is further updated by incorporating MINLP performance constraints, new device sizes, and new symmetry/matching constraints. A layout generator can then solve the updated layout

template to achieve a minimum layout area while meeting floorplan, symmetry/matching, proximity and performance constraints. The formulated layout-generation problem is solved by using graph-based techniques [60] combined with mixed-integer nonlinear programming.

5.3 Mathematical Modeling

5.3.1 Interconnect Parasitics

Layout parasitics considered in this chapter involve parasitic resistances and wire-substrate capacitances. A layout tile is a rectangle on a layer within a layout. Each circuit net includes a set of tiles (i.e., rectangles) and the total net-parasitic is the sum of its tile-parasitics. Assume the length and width of a tile are represented as (x_r-x_l) and (y_r-y_l), respectively, where (x_l, y_l) and (x_r, y_r) are left-bottom and right-top corner coordinates of the tile. The parasitic resistance and wire-substrate capacitance for a tile can be mathematically represented with its geometries as

$$R = \rho_{sh} \times (x_r-x_l) / (y_r-y_l). \tag{5.1}$$

$$C_{sub}=c_a \times (x_r-x_l) \times (y_r-y_l) + c_{sw} \times 2 \times (x_r-x_l). \tag{5.2}$$

where ρ_{sh} is the sheet resistance per unit length, c_a is a substrate capacitance per unit area, and c_{sw} is wire-substrate capacitance per unit length.

The analytical formulae above are applicable for lumped interconnect models. Very accurate interconnect RC estimations typically require the use of expensive *EM field solvers*. Even though these estimations are accurate, it might take days for *EM solvers* to

complete interconnect solving even for a medium-sized circuit. This method is obviously unacceptable for our layout problem since the maximum execution time of a layout process is within minutes as reported in this dissertation.

For analog design such as wire sizing and global routing, lumped interconnect models [59][61][62] are advocated due to their analytical closed-form expressions, fast computation speed, and good fidelity with respect to simulation. Lumped interconnect models typically underestimate the interconnect delay by a worst-case of 16% [61] for sub-micron layouts. This accuracy (i.e., $\geq$ 84%) is acceptable in our applications considering the significant reduction of interconnect solving time (i.e., from days to seconds). Thus, a simple lumped resistance-capacitance (RC) π-model as shown in Figure 4-3 is adopted in our approach to represent parasitic interconnects.

5.3.2 Performance Constraints

Performance sensitivities are calculated to represent the dependence of circuit performances with respect to interconnect parasitics. The product of a performance sensitivity and its corresponding parasitic (either resistance or capacitance) is used to model the performance deviation due to that parasitic. In addition, linear approximation is used to calculate the total performance deviation due to a set of parasitics that affects the same performance. In order to ensure certain desired circuit performance, .the total performance deviation must be restricted within a maximum allowed tolerance. Therefore, the sensitivity-based performance constraints can be represented as:

$$\sum S_{res} \times R\,(x,y) + \sum S_{subcap} \times C_{sub}(x,y) \leq \Delta W_{max}\,. \qquad (5.3)$$

where $R(x, y)$ and $C_{sub}(x, y)$ refer to geometric expressions for parasitic resistance and wire-substrate capacitance; S_{res} and S_{subcap} represent performance sensitivities for $R(x, y)$ and $C_{sub}(x, y)$; x and y represent related horizontal and vertical layout geometries.

5.3.3 Parasitic Matching Constraints

Matching parasitic constraints are indispensable for the parasitic-aware layout generation problem. Mismatch of parasitic structures can dramatically degrade the circuit performance. Thus, two parasitic geometric structures for a matching pair must be placed identically as shown in (5.4),

$$P_{n1}(x, y) = P_{n2}(x, y), \tag{5.4}$$

where $P_{n1}(x, y)$ and $P_{n2}(x, y)$ are two geometric parasitic expressions to be enforced as identical.

5.4 Sensitivity Computation

5.4.1 Central-Difference Sensitivity Approximation

As discussed in Section 4.3.2, central difference sensitivities can approximate the performance dependence on a parasitic around certain parasitic value as:

$$S_{ij} = [W_{ij}(P_{jUB} + \Delta) - W_{ij}(P_{jUB} - \Delta)] / 2\Delta, \tag{5.5}$$

where P_{jUB} is the upper bound of a parasitic P_j and Δ is a small interval. Here, the worst-case sensitivity is calculated around a parasitic upper bound due to: (1) a small parasitic

increase over its upper bound may result in a performance loss; (2) around an upper bound, the absolute value of performance sensitivity normally reaches the largest and therefore this is the worst case.

Less-sensitive and insensitive parasitics have smaller performance sensitivities which contribute merely a small portion of the total performance deviation. Thus, the central-difference approach can be used to generate plain worst-case sensitivities to approximate these parasitic impacts. However, for sensitive nets, the sensitivities are large and may significantly vary along with changing parasitics. Moreover, parasitic solutions of sensitive nets are usually small, far from their upper bounds around which the worst-case sensitivities are calculated. Thus, the worst-case sensitivities are insufficient to represent the impact of sensitive parasitics.

5.4.2 Piecewise Sensitivity Approximation

Piecewise sensitivities are modeled to better approximate the dependence of performance on sensitive parasitics. Firstly sensitivity analysis is conducted to identify a set of sensitive parasitics through simulations. Then, the feasible range for each sensitive parasitic is divided into a number of small segments. A feasible range is defined as the maximum scope that the value of a parasitic can cover. Worst-case performance sensitivity (called segmental sensitivity hereafter) is calculated to represent a parasitic impact for each segment, and piecewise sensitivity can be built up as a linear function of binary-integer variables and segmental sensitivities to represent the total performance dependence on a sensitive parasitic.

To avoid high computation cost due to applying abundant binary-integer variables, a proper number of segments needed for a sensitive parasitic should be determined considering solvability and efficiency. The number of segments for a sensitive parasitic is modeled as a function of two sensitivity-related factors (i.e., S_{jmax} and α) as show in (5.6) and (5.7). Here, the ratio (i.e., α) in (5.6) is calculated using the largest and smallest segmental sensitivities (i.e., S_{jmin} and S_{jmax} derived from simulations) for certain parasitic, to represent the fluctuation of its segmental sensitivities.

$$\alpha = S_{jmin} / S_{jmax}, \tag{5.6}$$

$$N_{seg} = N(\alpha, S_{jmax}), \tag{5.7}$$

where N_{seg} is the determined number of segments for a parasitic.

In this way, a reasonable accuracy can be achieved by moderately increasing the number of segments, whereas the unnecessary number of binary-integer variables for redundant segmental sensitivities is reduced to a minimum. It is to be noted that, piecewise sensitivities are only generated for sensitive parasitics, which is especially important when applying the piecewise scheme in handling a huge number of parasitics for larger designs. Moreover, it is not worthwhile to segment insensitive parasitics at all since these parasitics have tiny sensitivities and huge upper bounds that the related layout geometries can hardly achieve even without an optimization.

With a set of segmental sensitivities, we have adopted a binary-integer (0-1) method [63] to incorporate them into the formulation. In our formulation, 0-1 variables are applied to construct the piecewise sensitivity model as shown in (5.8), which is a linear function of a set of binary-integer variables limited by corresponding segmental

sensitivities. Binary-integer variables function as enable/disable switches for the selection of effective segmental sensitivity. Let $\{B_1, B_2, ..., B_N\}$ be a set of binary integers (i.e., B_n □$\{0, 1\}$, $1 \leq n \leq N$), and $\{S_1, S_2, ..., S_N\}$ as the corresponding segmental sensitivities for a sensitive parasitic, the piecewise scheme can be applied by incorporating the constraints as shown in (5.8)-(5.10), say, for a performance W_i with respect to a parasitic P_j.

$$S_{ij} = \sum_{n=1}^{N_{seg}} B_n S_n \quad , \tag{5.8}$$

$$\sum_{n=1}^{N_{res}} B_n = \sum_{n=1}^{N_{cap}} B_n = 1,\ B_n \in \{0,1\}, \tag{5.9}$$

$$B_n\,(P - P_{n1}) \geq 0 \ \&\&\ B_n\,(P - P_{n2}) \leq 0, \tag{5.10}$$

where N_{seg} refers to the determined number of segments for a parasitic, N_{res} / N_{cap} refers to the number of segments for a parasitic resistance/capacitance, and the domain (P_{n1}, P_{n2}) represents the segment where $B_n = 1$can hold.

For any iteration of optimization, only one binary variable in the set $\{B_1, B_2, ..., B_{Nseg}\}$ can be '1' whereas others are all 0's, as shown in (5.9). The set of $\{B_1, B_2, ..., B_{Nseg}\}$ corresponds to the set of parasitic segments where $\{S_1, S_2, ..., S_{Nseg}\}$ is calculated. Thus, only one effective segmental sensitivity in $\{S_1, S_2, ..., S_{Nseg}\}$ is enabled in the optimization. Besides, we must ensure that an effective segmental sensitivity is enabled only when a parasitic is accordingly bounded within that segment. This bounding is applied using (5.10), where B_n should be a '1' if and only if both $(1 \times (P - P_{n1}) \geq 0)$ and $(1 \times (P - P_{n2}) \leq 0)$ hold (i.e., $P_{n1} \leq P \leq P_{n2}$). At the same time, (5.10) is an invariant 'TRUE' when B_n is '0'. By incorporating the piecewise sensitivities as shown in (5.8), the piecewise-based performance constraints are represented as shown in (5.11).

$$\sum_{j=1}^{N}[(\sum_{n=1}^{N_{seg}} B_n S_n) \times P_j(x,y)] \le \Delta W_{max} \ , \tag{5.11}$$

where N refers to the number of parasitics affecting a considered performance, and N_{seg} refers to the determined number of segments for a considered parasitic.

5.5 Problem Formulation and Layout Solving

5.5.1 Problem Formulation

The performance-constrained piecewise-based parasitic problem is formulated as shown in (5.12)-(5.16), besides basic constraints for design rules and symmetry.

$$\textit{Minimize} \quad (x_{rr} - x_{ll}) \times (y_{rr} - y_{ll}), \tag{5.12}$$

Subject To

$$\sum_{j=1}^{N}[(\sum_{n=1}^{N_{seg}} B_n S_n) \times R_j(x,y) + (\sum_{n=1}^{N_{seg}} B_n S_n) \times C_j(x,y)] \le \Delta W_{i\_max} \ , \tag{5.13}$$

$$P_{m1}(x,y) = P_{m2}(x,y), \tag{5.14}$$

$$\sum_{n=1}^{N_{seg}} B_n = \sum_{n=1}^{N_{seg}} B_n = 1, \ B_n \in \{0,1\}, \tag{5.15}$$

$$B_n\,(P - P_{n1}) \ge 0 \ \&\& \ B_n\,(P - P_{n2}) \le 0, \tag{5.16}$$

where (x_{rr}, x_{ll}, y_{rr}, y_{ll}) represents the boundaries of a layout, $\sum_{n=1}^{N_{seg}} B_n S_n$ and $\sum_{n=1}^{N_{seg}} B_n S_n$ refer to the piecewise performance sensitivities with respect to parasitic resistances and wire-substrate capacitances, respectively.

Rather than generating a set of geometric constraints with parasitic bounds, the intrinsic relationship between performance and parasitics is modeled as a mixed-integer nonlinear function of layout geometries restricted within the maximum allowed performance deviation. Besides the performance constraints due to individual parasitics, matching parasitic constraints as shown in (5.14) are also included in the constraint set to avoid undesirable circuit performance loss.

Due to the nonlinearity of parasitic expressions as shown in (5.1)-(5.2) and binary-integer features of performance constraints as shown in (5.13), the formulated problem is a mixed-integer nonlinear programming problem [64]. IPOPT [49] is selected as our MINLP solver due to the following facts. IPOPT can solve nonlinear problems with non-convex constraints involving binary-integer (0-1) design variables. Moreover, it not only features promising local convergence for nonlinear search, but also guarantees global convergence by using a filter-based line search strategy.

5.5.2 Layout Generation

The layout generation flow is shown in Figure 5-2. The layout generation includes three steps: (1) generation of a reduced graph from symmetry/matching and performance constraints for a layout; (2) solving the reduced graph with MINLP; (3) solving the complete LP constraint graph with longest-path algorithm. It is to be noted that, horizontal and vertical constraint graphs are generated and solved separately within the flow. To improve the searching efficiency of the MINLP solver, a two-phase solving scheme is utilized here. In the first phase, a MINLP-only problem of a reduced graph (i.e.,

considering only performance and symmetry/matching constraints) is solved using mixed-integer nonlinear programming. In the second phase, a linear-only compaction of an LP complete constraint graph is conducted using graph-based optimization combined with linear programming. The solutions out of the first phase provide the start points of the second phase to separate the searching effort from a complete MINLP to a complete LP.

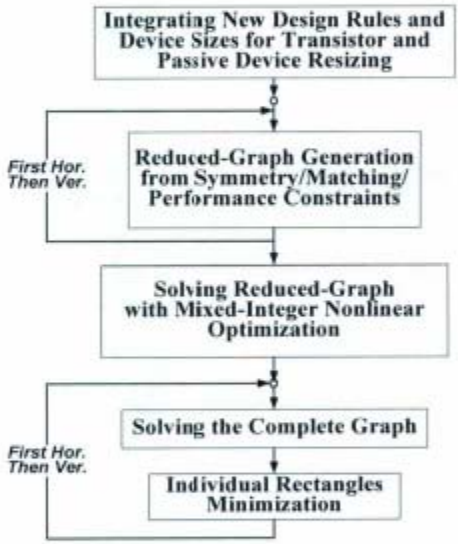


Figure 5-2 A performance-driven layout generation flow.

With the entire set of constraints, layout generation is conducted to solve the layout problem. The layout generator is a template-solving engine that enforces the constraints related to extracted expertise from original layouts, symmetry/matching, performance

requirements, new device sizes and new technology design rules. To enhance the computation speed, graph compaction techniques are adopted to convert linear equations or inequalities into a graph form. A reduced graph is a simplified graph which is equivalent (for the core nodes) to the original constraint graph but with many fewer nodes and arcs. A small parasitic-included laycut is shown in Figure 5-3 as an example, and its graph-based layout-generation steps are shown in Figure 5-4.

An original graph as shown in Figure 5-4(a) can be derived by converting all linear constraints (e.g., *minimum-width* constraints) in Figure 5-3 into a graph form. However, since MINLP performance constraints (e.g., the constraint in Figure 5-4(b-1) or Figure 5-4(d-1)) or symmetry/matching constraints (e.g., the constraint in Figure 5-4(b-2) or Figure 5-4(d-2)) are not allowed in graph techniques, these constraints are kept as equations/inequalities which are solved together with the reduced graph.

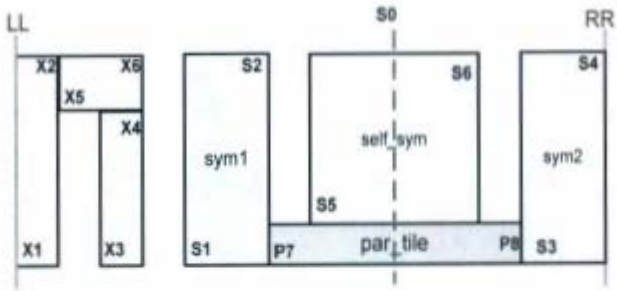


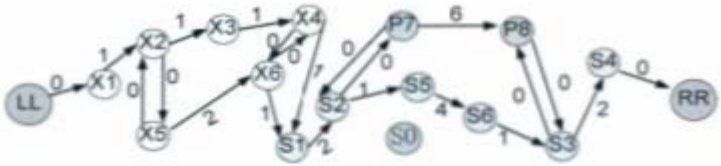
Figure 5-3 An example layout with parasitic tiles.

Figure 5-4(a) shows an original constraint graph consisting of 17 nodes and 22 arcs. In the reduced graph of Figure 5-4(c), only 11 nodes (called core nodes that are marked

gray in the figures) and 12 arcs related to boundary, symmetry/matching and performance are involved. To ensure the equivalence between the original graph and the reduced one, weights of arcs between nodes are calculated by applying any longest-path algorithm (e.g., Bellman-Ford algorithm in this work) from each core node to the other core nodes in the reduced graph. For example, the arc weight from *P7* to *P8* has a longest path of 6 that is derived from the original graph. The core graph with longest-path arc weights is then converted to a set of equations and/or inequalities. By combining the converted linear constraints with performance and symmetry constraints, the complete reduced-graph constraint set is then constructed as shown in Figure 5-4(c) and Figure 5-4(d).

To form a complete LP graph that includes all needed constraints, we must incorporate Figure 5-4(d) to the original graph. Thus, these 4-parameter and MINLP constraints must be converted into combinations of two-variable constrained equations. To facilitate this conversion, the optimum distances for symmetry/matching constraints and the solutions of performance constraints must be obtained by solving the reduced-graph constraint set which includes the constraints of both Figure 5-4(c) and Figure 5-4(d).

After a set of optimal solutions is exported by the MINLP solver, the solutions related to symmetry/matching and performance constraints construct the optimal arc weights for the 2-variable converted performance/symmetry/matching constraints as shown in Figure 5-4(e). By incorporating these optimal arc weights into the original graph, a complete graph is formed as shown in Figure 5-4(f) in a complete LP form. This complete constraint graph can be solved using any longest-path algorithm. The solutions derived from this constraint graph are mapped back to specify a target layout.

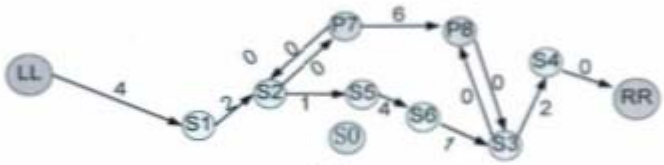


(a)

$$(T_1 \times S_1 + T_2 \times S_2) \times RES(par\_tile) = (T_1 \times S_1 + T_2 \times S_2) \times \rho_{s,t} \frac{(P8 - P7)}{(W8 - W7)} \le \Delta W_{max} \quad \text{(b-1)}$$

S6-S0 = S0-S5 S3-S0 = S0-S2 P8-S0 = S0-P7 S4-S3 = S2-S1 (b-2)

(b)



(c)

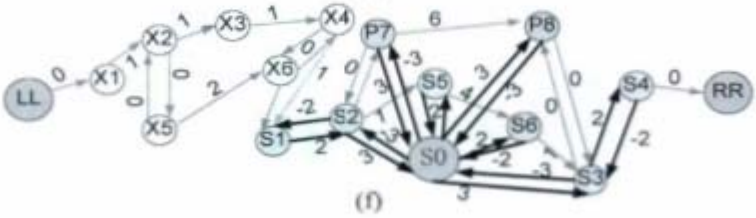
$$(T_1 \times S_1 + T_2 \times S_2) \times RES(par\_tile) = (T_1 \times S_1 + T_2 \times S_2) \times \rho_{s,t} \frac{(P8 - P7)}{(W8 - W7)} \le \Delta W_{max} \quad \text{(d-1)}$$

S6-S0 = S0-S5 S3-S0 = S0-S2 P8-S0 = S0-P7 S4-S3 = S2-S1 (d-2)

(d)

S6-S0 = S0-S5 =2 S4-S3 = S2-S1 =2 S3-S0 = S0-S2 = 3
P8-S0 = S0-P7 = 3 P8-P7 = 6

(e)



(f)

Figure 5-4 An example of graph-based layout solving, (a) an original graph, (c) a reduced-sized equivalent core graph, (f) a complete graph.

5.6 Experimental Results

The proposed algorithm has been integrated into IPRAIL. In this section, we report results of layout retargeting on two analog circuits: a *two-stage Miller-compensated operational amplifier* as shown in Figure 4-4 and *a single-ended folded cascode opamp* as shown in Figure 4-5. The retargeting was conducted from a 0.25μm CMOS process to a 0.18μm CMOS process with updated performance specifications.

Through a number of simulations, sensitivity analysis was first conducted to identify sensitive individual and matching parasitics. For clarity, the critical parasitics are indicated in the circuit schematics. The RC π-model presented in Figure 4-3 is used in the circuit simulations. For the two-stage opamp, individual parasitics of *n3b*, *n2a*, and *n2c* were found sensitive and a sensitive matching parasitic of *n3a* & *n3c* was detected, as depicted in Figure 4-4. For the cascode opamp in Figure 4-5, sensitive matching parasitics of *n3b* & *n3c*, *n1c* & *n4c*, *n1b* & *n4b* and *n2* & *n6* were detected.

After the setup of piecewise performance sensitivities, parasitic-aware layout retargeting was performed using our proposed performance-constrained mixed-integer method (called PMI). To demonstrate the superior effectiveness and efficiency of our method, bound-based parasitic-aware retargeting [9] (called PB) was also conducted. In addition, to show that piecewise sensitivities obtain higher accuracy over simple central-difference worst-case sensitivities, we set up a similar flow (called PS) of using single worst-case sensitivities (non-piecewise) supported by nonlinear programming. All these methods were used for retargeting the same layouts and the results are compared.

Performances of AC gain, bandwidth, phase margin, and gain margin were considered in retargeting. Here, to determine the number of segments for sensitive parasitics with a good trade-off, criteria tables were designed as shown in Table 5-1 for parasitic resistances and Table 5-2 for parasitic capacitances. In Table 5-1 and Table 5-2, the first two columns report the criterion for the maximum segmental sensitivities (i.e., $|S_{i\_max}|$) and the fluctuation ratios (i.e., $|\alpha|$) with respect to parasitic resistances and wire-substrate capacitances, respectively. The last column lists the determined number of segments for a considered parasitic. For instance, in the fourth row of Table 5-1, [0.6, 0.8) is a domain from 0.6 to 0.8 including the value 0.6 but excluding the value 0.8. When the worst-case segmental sensitivity for a parasitic resistance is within (0.025, 0.05) and its fluctuation is within [0.6, 0.8), the number of segments for this parasitic is determined as 2. An observation is that, the criterion for wire-substrate capacitances are much easier than those of parasitic resistances. This is because, the performance sensitivities with respect to wire-substrate capacitances are generally very small for analog retargeting at lower frequency, compared to those for parasitic resistances.

Parasitic bounds for PB and performance sensitivities for PS/PMI are listed in Table 5-3 for the two-stage opamp and Table 5-4 for the cascode opamp. For example, for the *R3b* in Table 5-3, we have $|S_{i\_max}| = 0.068 \in [0.05, 0.1)$ and $|\alpha| = 0.63 \in [0.6, 0.7)$. The number of segments for the parasitic of *n3b* is then determined as 3 according to the criteria in the eighth row of Table 5-1. Due to space limitation, information of only sensitive parasitic resistances regarding AC gain for the two-stage opamp and information

of only sensitive wire-substrate capacitances regarding phase margin for the cascode opamp are reported.

Table 5-1 Criterion of determining the number of segments for parasitic resistances.

$\lvert S_{i\_max}\rvert$	$\lvert\alpha\rvert$	N_{seg}
≤ 0.025	Any	1
Any	[0.8, 1]	1
(0.025, 0.05)	[0.6, 0.8)	2
	[0.4, 0.6)	3
	[0.025, 0.4)	4
[0.05, 0.1)	[0.7, 0.8)	2
	[0.6, 0.7)	3
	[0.05, 0.6)	4
≥ 0.1	Any	4
Any	≤ 0.2	4

Table 5-2 Criterion of determining the number of segments for parasitic capacitances.

$\lvert S_{i\_max}\rvert$	$\lvert\alpha\rvert$	N_{seg}
≤ 0.001	Any	1
Any	≤ 0.5	4
≥ 0.001	[0.9, 1]	1
	[0.7, 0.9)	2
	(0.5, 0.7)	3

In Table 5-3 and Table 5-4, the PB bound and PS worst-case sensitivities are listed in the second and third columns. The last column reports the PMI piecewise sensitivities, where distinct 0-1 variables (i.e., B_i) are assigned to enforce effective segmental sensitivities. For instance, sensitive parasitic resistance *R2c* has a PB bound of 19 Ω, a PS worst-case sensitivity (regarding AC gain) of -0.14 dB/Ω, and a PMI piecewise sensitivities of {-0.115, -0.128, -0.132, -0.145} dB/ Ω corresponding to the set of parasitic

segments {(0, 4.75 Ω), (4.75 Ω, 9.5 Ω), (9.5 Ω, 14.25 Ω), (14.25 Ω, 19.0 Ω)} respectively. Moreover, parasitics within a matching pair (as one optimization object in our formulation) are forced to share the same set of 0-1 variables and segmental sensitivities to ensure the same selection of an effective segmental sensitivity during optimization (e.g., *R3a* & *R3c* share $\{B_1, B_2\}$).

Table 5-3 Selected PB parasitic bounds and PS/PMI sensitivities of AC gain with respect to parasitic resistances for the two-stage opamp.

Res. (Ω)	PB-Bound (Ω)	PS Sensitivity (dB/Ω)	PMI-Piecewise Sensitivities (dB/Ω) (B_i: 0-1 variable)
R3a	4.1	-0.032	$-0.009B_0-0.022B_1-0.032B_2$
R3c	4.1	-0.032	$-0.022B_1-0.032B_2$
R3b	29.4	-0.068	$-0.043B_3-0.052B_4-0.068B_5$
R2a	24.0	0.031	$0.031B_6$
R2b	50.2	-0.001	$-0.001B_7$
R2c	19	-0.14	$-0.115B_8-0.128B_9-0.132B_{10}-0.145B_{11}$
R2e	112.9	0.001	$0.001B_{12}$

Table 5-4 Selected PB parasitic bounds and PS/PMI sensitivities of phase margin with respect to parasitic capacitances for the cascode opamp.

Cap.	PB Bound (fF)	PS Sensitivity (degree/ fF)	PMI-Piecewise Sensitivities (degree/fF) (B_i: 0-1 variable)
C1b	2.7	-0.043	$-0.035B_1-0.043B_2$
C4b	2.7	-0.043	$-0.035B_1-0.043B_2$
C1c	2.7	-0.043	$-0.038B_3-0.043B_4$
C4c	2.7	-0.043	$-0.038B_3-0.043B_4$
C3b	17.1	-0.008	$-0.005B_5-0.007B_6-0.008B_7$
C3c	17.1	-0.008	$-0.005B_5-0.007B_6-0.008B_7$
C2	8.0	0.0014	$0.003B_6+0.021B_7+0.005B_8+0.$
C6	8.0	0.0014	$0.003B_6+0.021B_7+0.005B_8+0.$

After each retargeting, parasitics of the target layout were extracted and these values are reported in Table 5-5 and Table 5-6. From these tables, we observe that, most of the extracted parasitics by PS are not around their upper bounds where their worst-case sensitivities are calculated whereas all the extracted parasitics by PMI fall into their expected segments. For instance, in Table 5-5, PS sensitivity of AC gain to *R2c* is calculated within a tiny domain of (18.9 Ω, 19.0 Ω) around its upper bound, but its extracted parasitic is only 3.43 Ω, which is far away from its supposed segment where its sensitivity is calculated. On the other hand, its PMI extracted parasitic of 2.07 Ω falls into the expected segment of (0 Ω, 4.75 Ω). Similarly, in Table 5-6, the extracted parasitic of a matching pair *C2 & C6* reaches 1.1 fF for PS, which is far outside its sensitivity-calculation domain of (7.9 fF, 8.0 fF), but the extracted parasitic from PMI is 2.7 fF that fits the right segment of (2.0 fF, 4.0 fF). It is worth mentioning that, the parasitic bounds generated by using PB method (from handcrafted simulations) may not be equal to their counterpart bounds generated using PMI (using the proposed bisearch algorithm as shown in Figure 4-1). Thus, the extracted parasitics for PMI retargeting are not required to be restricted by the PB bounds. For example, the matching parasitic of *R3a & R3c* in Table 5-5 achieves an extracted resistance of 6.09 Ω & 6.02 Ω. These two values are both larger than the PB bound of 3.98, but they are less than the PMI bound of 10.8 Ω.

Post-layout simulations were conducted and the simulated performances for distinct methods are summarized in Table 5-7. The target layouts by PMI have smaller area than those of PB or PS (by 8.5% and 3.6% for the two-stage opamp, as well as 5.6% and 3.1% for the cascode opamp, respectively). By applying piecewise sensitivities in the PMI formulation, performance constraints are correctly enforced in the optimization.

Therefore, better circuit performances were achieved for PMI compared to PB or PS as reported in Table 5-7. For example, PMI obtained a better AC gain of 64.3dB compared to 64.0dB by PS for the two-stage opamp. This improvement is strongly related to the smaller degradation (i.e., product of parasitic value and sensitivity) caused by a smaller extracted parasitic (i.e., 2.07 Ω) of *R2c* and its smaller effective segmental sensitivity in absolute value (i.e., -0.115 dB/Ω) by PMI, compared to the extracted parasitics obtained by PS (i.e., 3.43 Ω and -0.145 dB/Ω). Similarly, phase margin is improved in the cascode opamp due to the accurate extracted parasitic and effective segmental sensitivity of a matching parasitic *C2 & C6*. As reported in Table 5-7, PMI layouts achieved the best circuit performances with the least layout areas over PB or PS although all three methods can manage the performances above the baseline of specifications. The original and target layouts generated by PMI are depicted in Figure 5-5 for the two-stage opamp and Figure 5-6 for the cascode opamp. In the target layout of Figure 5-5(b), the interconnect wires, especially for the differential pair of nets *n3a & n3c*, are obviously thinned compared to those in the original layout of Figure 5-5(a). This observation is due to the larger extracted *R3a & R3c* by PMI (i.e., thinner wires of fixed length have larger resistances) as shown in Table 5-5, as compared to the extracted *R3a & R3c* by PB. These larger extracted parasitics (while still meeting performance goals) provides loosened parasitic requirements for the retargeting problem, which in turn widens the solving flexibility of the layout generator.

The execution times (running on a *Sun Blade 100* workstation) of PB/PS/PMI are summarized in Table 5-8 for both layouts. The layout generation times spent by PS and PMI were less than those of PB on both opamps. In particular, the parasitic solving time

by PMI is 22% less than that of PB for the cascode opamp, and 6% less for the two-stage opamp. For both circuits, PMI performs layout retargeting within 3 minutes of CPU time. With identical number of nodes and parasitic tiles on the same layouts, the PMI method is able to significantly reduce the execution time compared to PB method but effectively manage analog retargeting. Moreover, modest time increase is needed by PMI compared to PS as a trade-off due to its improvement on performance and layout area.

Table 5-5 Extracted parasitics from the target two-stage opamp layouts by PB, PS, and PMI, and related PS/PMI parasitic segments.

Res (Ω)	PB	PS	Sensitivity-Range (PS)	PMI	Sensitivity-Segments (PMI)
R3a	3.98	3.91	(4.0, 4.1)	6.09	(5.4, 10.8)
R3c	3.98	3.91	(4.0, 4.1)	6.02	(5.4, 10.8)
R3b	6.10	3.98	(29.3, 29.4)	6.61	(0.1, 9.8)
R2a	2.78	2.79	(23.9, 24.0)	2.73	(0, 24)
R2b	2.52	2.46	(50.1, 50.2)	2.35	(0, 50.2)
R2c	3.94	3.43	(18.9, 19.0)	2.07	(0, 4.75)
R2e	3.75	3.90	(112.8, 112.9)	3.90	(0, 112.9)

Table 5-6 Extracted parasitics from the target cascode opamp layouts by PB, PS, and PMI, and related PS/PMI parasitic segments.

Cap	PB	PS	Sensitivity-Range (PS)	PMI	Sensitivity-Segments (PMI)
C1b	0.35	0.33	(2.6, 2.7)	0.33	(0, 1.35)
C4b	0.35	0.33	(2.6, 2.7)	0.33	(0, 1.35)
C1c	0.42	0.42	(2.6, 2.7)	0.42	(0, 1.35)
C4c	0.42	0.42	(2.6, 2.7)	0.42	(0, 1.35)
C3b	2.3	2.0	(17.0, 17.1)	2.1	(0, 5.7)
C3c	2.3	2.0	(17.0, 17.1)	2.1	(0, 5.7)
C2	1.9	1.1	(7.9, 8.0)	2.7	(2.0, 4.0)
C6	1.9	1.1	(7.9, 8.0)	2.7	(2.0, 4.0)

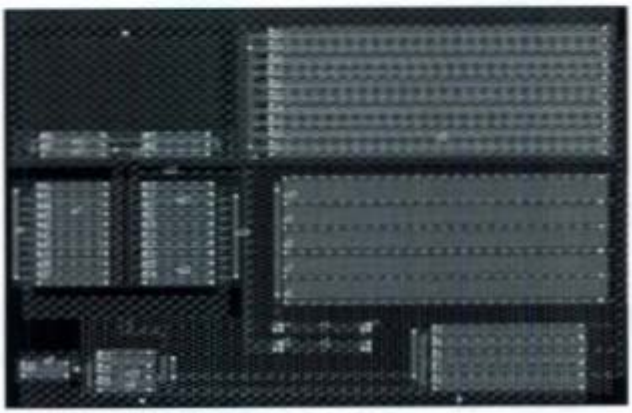
For a regular analog design, the circuit size is normally not very big (i.e., dozens of transistors or devices). However, the layout time may be more than tens of hours if designers manually manage the layout retargeting to fit a small-feature technology process. Moreover, it is normally difficult to ensure the satisfaction of performance for the retargeted layout. Therefore, the proposed algorithm, which is to automate the layout retargeting with the full consideration of parasitics, is worthwhile for the analog layout design. Furthermore, a large-size complicated analog design normally deploys a hierarchical design paradigm, which divides the entire circuit to several medium-size blocks and focuses on the layout of each block before integration. Due to the iterative calls of design automation for small or medium-size analog blocks, the time efficiency improvement gained from this thesis would be influential to the large-size complicated analog design.

Table 5-7 Results of post-layout simulations for the target layouts.

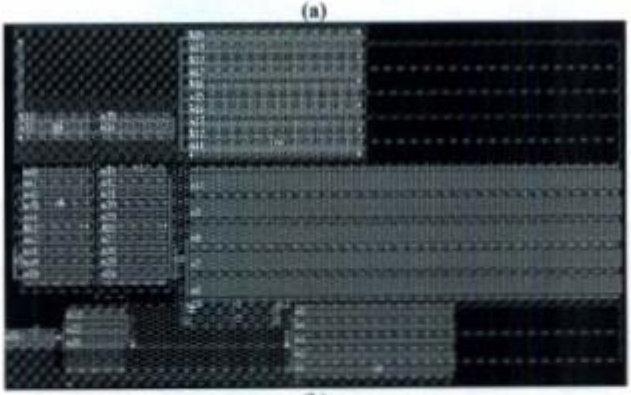
		Gain (dB)	BW (MHz)	PM (°)	GM (dB)	Area (μm^2)
Two-stage opamp	Specification	60.0	100	90.0	10.0	-
	Ideal(no parasitics)	64.5	107.3	90.5	16.8	-
	PB	63.9	105.1	90.5	16.7	3084
	PS	64.0	105.0	90.4	16.7	2920
	PMI	64.3	106.4	90.5	16.7	2815
Folded-cascode opamp	Specification	60.0	60.0	60.0	10.0	-
	Ideal(no parasitics)	60.7	63.7	61.2	10.4	-
	PB	60.6	63.7	60.3	10.2	2320
	PS	60.6	63.4	60.6	10.4	2262
	PMI	60.6	63.7	61.1	10.4	2190

Table 5-8 Summary of time efficiency for PB/PS/PMI.

	Two-Stage Opamp			Cascode Opamp		
	PB	PS	PMI	PB	PS	PMI
#Nodes	1050			1103		
#Parasitic Tiles	26			38		
Template Extraction	16.5s	16.5s	16.5s	20.4s	20.4s	20.4s
Parasitic Solving	42.7s	36.1s	40.3s	131.4s	69.3s	99.2s
Layout Generation	69.9s	63.3s	67.5s	212.0s	149.9s	179.8s

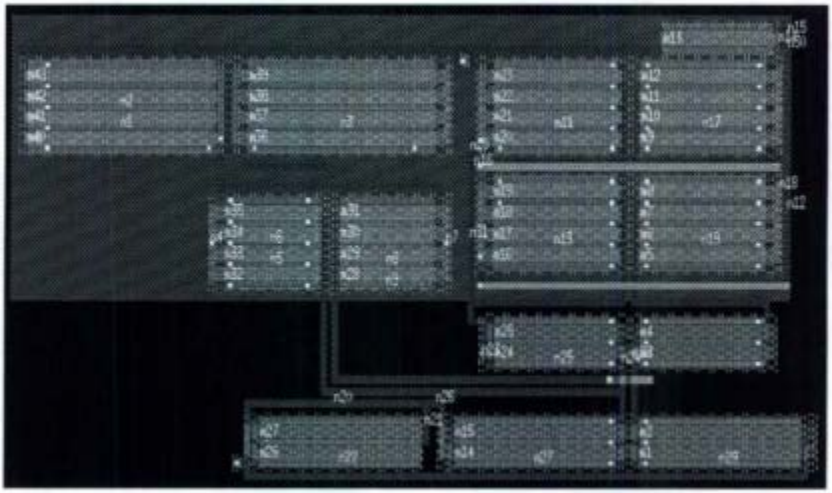


(a)

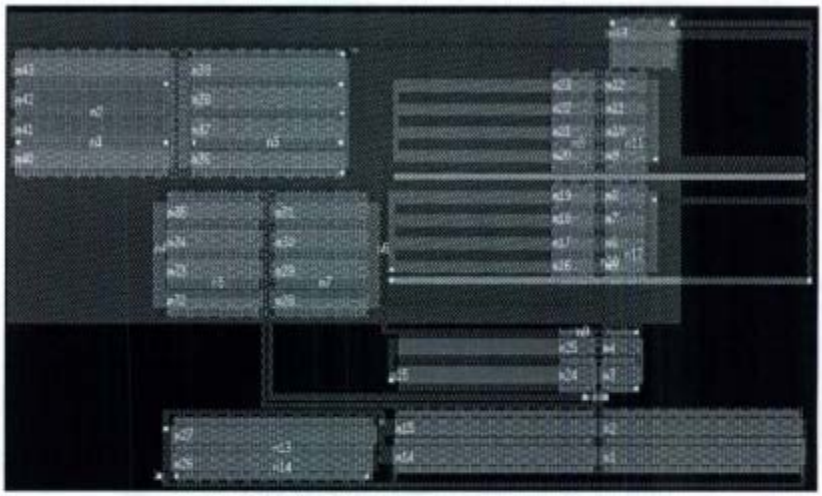


(b)

Figure 5-5 (a) Original and (b) target layouts of the two-stage opamp by PMI.



(a)



(b)

Figure 5-6 (a) Original and (b) target layouts of the cascode opamp by PMI.

5.7 Summary

In this chapter, a performance-constrained parasitic-aware layout retargeting algorithm was presented. Different from the conventional sensitivity analysis, we deployed central-difference approximation that uses any simulators for segmental sensitivity computation. A piecewise sensitivity model was designed to quantify the performance dependence with respect to a sensitive parasitic with improved accuracy. Moreover, mixed-integer performance constraints due to parasitics were included in the formulated MINLP problem rather than through indirect parasitic-bound constraints. Graph techniques and mixed-integer nonlinear programming are effectively combined to solve the formulated parasitic-aware problem. Our experimental results show that the proposed algorithm achieves effective retargeting of analog circuits with smaller layout area and significant reduction of execution time, as compared to the alternative methods.

However, inductive impacts and wire-coupling capacitances were not involved in the formulation, which inevitably degrades its effectiveness for layout retargeting at higher/RF frequency. Due to the growing operation frequency, the parasitic inductive and wire-coupling capacitive impacts become more significant in affecting RF circuit performance. Thus, inductive and wire-coupling capacitive parasitics must be controlled to avoid malfunction of RF designs if only considering parasitic resistances and wire-substrate capacitances otherwise. For this concern, an RF retargeting problem is handled in Chapter 6 considering complete RLC parasitics.

6. Layout Retargeting for Radio Frequency Integrated Circuits

Resistive, inductive, and capacitive (i.e., *RLC*) parasitics are all significant for high-performance RF designs. At lower frequency, the resistive parasitic is the major contributing factor to the parasitic-induced performance degradation, compared to its capacitive and inductive counterparts. However, at radio frequency, RF performance becomes very sensitive to inductive and wire-coupling parasitics. These parasitics must be considered in parasitic-aware RF layout optimization/retargeting [5], besides the parasitic resistances and the wire-substrate capacitances that are typically considered in analog optimization. The proposed performance-constrained RF retargeting algorithm has been demonstrated to be effective and efficient for retargeting RF layouts with updated performance specifications.

6.1 Introduction

With higher clock frequencies and faster transistor rise/fall time in modern VLSI circuits, long-signal wires exhibit transmission line effects [65]. Currently, inductances have become more significant for high-speed VLSI circuits because of the following VLSI trends [66]: First, development of lower permeability dielectrics increases the relative importance of parasitic inductances compared to that of parasitic resistances. Second, longer interconnects are advocated in the modern complex RF layouts and hence wire-coupling capacitances and interconnect inductances become considerably larger. For

instance, the wire-coupling capacitance at radio frequency can be much larger than its wire-substrate counterpart due to the increasingly dense wiring structures. Moreover, the mutual inductance for a pair of wires can be comparable to its self inductance, due to the advocated longer-and-thinner wire style and the shrunk spacing. The parasitic inductance for an interconnect wire, is the sum of its self and induced mutual inductances. Therefore, interconnect parasitics of self inductances, mutual inductances and wire-coupling capacitances must be optimized in RF retargeting in addition to the resistive and wire-substrate parasitics, to ensure desired RF circuit performance.

The rest of the chapter is organized as follows. Section 6.2 introduces the proposed RLC interconnect model as well as the applicable RLC formulae. Section 6.3 discusses the extraction and calculation of mutual inductances. The extraction of wire-coupling capacitances is detailed in Section 6.4. The RF retargeting algorithm is introduced in Section 6.5. The experimental results are reported in Section 6.6 followed by a brief summary drawn in Section 6.7.

6.2 An RLC Interconnect Model

Inductive components can be excluded from the interconnect representation when circuits operate at lower frequency because the inductive impact on performance is far smaller than its resistive counterpart. However, in the high-frequency domain, parasitic inductances may have comparable or even larger impact on circuit performance.

In deep-submicron high-frequency IC designs [39], simple RC interconnect modeling ignoring inductive components can bring errors of up to 30% in the total

propagation delay compared to complete RLC interconnect modeling. Due to the advocated lower-resistance interconnect lines, the reactive component of wire impedance becomes comparable to its resistive counterpart. Moreover, the mutual inductive impacts become larger due to complex wire-coupling structures with longer return current paths.

In order to obtain quick evaluation of RLC parasitics without compromising accuracy, a lumped RLC interconnect π-model as shown in Figure 6-1 is adopted to incorporate an inductive component into the traditional RC π-model. Within this model, self inductance is computed by layout geometries besides net-based parasitic resistance and capacitances [65] [67]. Each circuit net includes a set of tiles (i.e., rectangles) and the total net parasitic is the sum of its tile parasitics. The parasitic resistances, wire-substrate and wire-coupling capacitances are calculated using layout geometries of a tile as shown in (6.1) - (6.3) [9], while long-wire self-inductance is calculated with (6.4) [65]. The mutual inductance formulae are introduced in Section 6.3.1 since mutual inductances have very complex formulae for different scenarios.

$$R = \rho_{sh} \times (x_r - x_l) / (y_r - y_l), \tag{6.1}$$

$$C_{sub} = c_a \times (x_r - x_l) \times (y_r - y_l) + c_{sw} \times 2 \times (x_r - x_l), \tag{6.2}$$

$$C_{coup} = c_c \times (x_r - x_l) / distance, \tag{6.3}$$

$$L_{self} = \frac{\mu}{2\pi}[l \times \ln(\frac{2l}{w}) + 0.5 + \frac{0.23w}{l}], \tag{6.4}$$

where μ is the technology-specific inter-layer material permeability, c_c is coupling capacitance per unit length , and l and w are the length and width of a tile on a layer.

The thickness of a layout tile (e.g. 0.53 μm for *TSMC18*) is ignored as it is far smaller than the length of a tile that causes an appreciable inductance. The typical width of a small-sized RF layout is mostly over 100 μm, and a noticeable inductance can arise when the length of a wire is over 10 μm. The thickness of 0.53 μm is only around 1/20 of 10um. Thus, the thickness is not worth handling in our RF retargeting considering its added 3D solving effort.

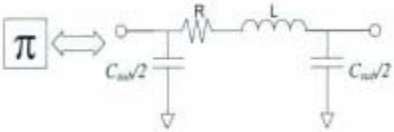


Figure 6-1 An RLC Interconnect π-model.

6.3 Mutual Inductances

6.3.1 Mutual Inductance Calculation

Due to the growing operating frequency, interconnects in high-speed ICs exhibit transmission-line effects, which make the analytical inductance calculation harder. There are three available calculation techniques for on-chip inductances [32]: *EM solvers*, segmented interconnect solving, and lumped interconnect solving.

The *EM solvers* [68] apply the numerical solutions of *Maxwell's* equations. This is the most accurate approach to evaluate 3D inductances in multilayer RF layouts. However, these simulations may take days or even weeks to complete, which is unpractical for our RF retargeting, even though they are accurate. Moreover, these

simulations require designers' expertise of using specific simulators because detailed process information of the used technology is usually unavailable.

Another approach is to model a mutual inductance as a transmission line network and use a distributed parasitic model [69][70]. Mutual inductance can be calculated by assembling a set of segmental inductances. However, due to the complexity of wiring-coupling structures, even each segmental inductance has a very complex inductance expression. Thus, these segmented calculation schemes of mutual inductances still significantly upgrades the size and complexity of interconnect evaluation, which leads to unacceptable solving time of the RF layout retargeting problem.

The third approach is to approximate mutual inductances in a lumped RLC model with applicable analytical formulae for distinct parallel-wire scenarios [65][67]. In [67], formulae are developed for 29 parallel-wire scenarios (i.e., different relative positions for a pair of wires). However, the classification criterion is so dense that, unacceptable execution time may be caused for a nonlinear-programming solver to simultaneously handle 29 complex nonlinear functions. In [65], mutual inductances are approximated for 7 cases of wiring structure as shown in Figure 6-2. Distinct formulae are developed for fast evaluation of appreciable mutual inductances without compromising accuracy. Thus, these formulae are generally adopted for evaluating mutual inductances with a good trade-off between efficiency and accuracy. For instance, (6.5) is used to estimate mutual inductance for two parallel interconnect wires with equal length,

$$M = \frac{\mu}{2\pi}[l \times \ln(\frac{2l}{d}) - l + d] \quad , \tag{6.5}$$

where d is the separation between two wires and l is the wire length.

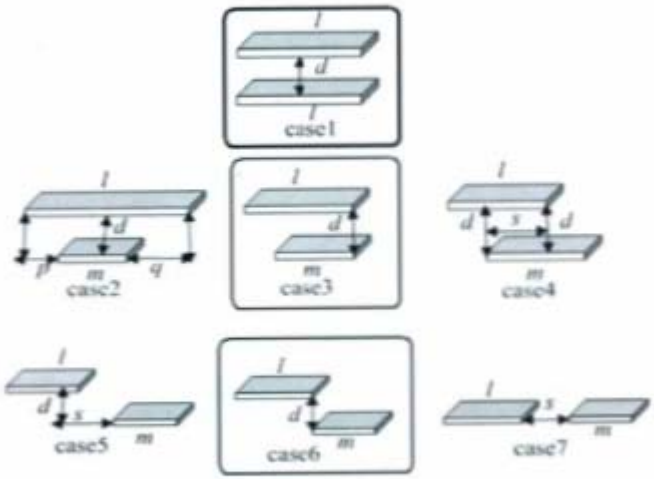


Figure 6-2 Complex parallel-wire structures with equal/unequal length.

However, RF layouts may have considerable parallel wires with unequal length. Moreover, a net interconnect can have several cascaded segments (tiles) in sequence. The mutual inductance for such a net must be calculated considering all its segments. In cases 2-7 of Figure 6-2, the mutual inductance for an unequal-length pair is calculated as an algebraic sum of the mutual inductances for its segments. Each segment is treated as an equal-length pair and its segmental mutual inductance is estimated using (6.5). Thus, a mutual-inductance formula for two parallel wires with unequal length can be constructed by calculating all its segmental mutual inductances with (6.5). For instance, the mutual inductance formula for case 2 is constructed as shown in (6.6).

$$\mathrm{M} = \frac{1}{2}[(M_{m+p} + M_{m+q}) - (M_p + M_q)] \ , \tag{6.6}$$

where M_{m+p} refers to the mutual inductance for an equal-length segment $(m+p)$, and M_{m+q} refers to the mutual inductance for an equal-length segment $(m+q)$. By expanding M_{m+p}, M_{m+q}, M_p and M_q using (6.5), the mutual inductance of (6.6) can be represented with tile geometries as shown in (6.7),

$$\mathrm{M2} = \frac{\mu}{4\pi}[(m \times \ln(\frac{4(m+q)(m+p)}{d^2}) + p \times \ln(\frac{m+p}{p}) + q \times \ln(\frac{m+q}{q}) - 2m] \ , \tag{6.7}$$

where l, m, d, p and q are related symmetry parameters as indicated in Figure 6-2. Similarly, the formulae for cases 3-7 can be derived as shown in (6.8) - (6.11).

$$\mathrm{M3} = \frac{\mu}{4\pi}[l \times \ln(\frac{l}{l-m}) + m \times \ln(\frac{4m(l-m)}{d^2}) - 2m + d] \tag{6.8}$$

$$\mathrm{M4} = \frac{\mu}{4\pi}[(l-s) \times \ln(\frac{l+m-s}{l-s}) + m \times \ln(\frac{l+m-s}{m-s}) + s \times \ln(\frac{4s(m-s)}{d^2}) - 2s] \tag{6.9}$$

$$\mathrm{M5} = \mathrm{M7} = \frac{\mu}{4\pi}[(l+s) \times \ln(\frac{l+m+s}{l+s}) + m \times \ln(\frac{l+m+s}{m+s}) + s \times \ln(\frac{s}{m+s})] \tag{6.10}$$

$$\mathrm{M6} = \frac{\mu}{4\pi}[l \times \ln(\frac{l+m}{l}) + m \times \ln(\frac{l+m}{m}) - d] \tag{6.11}$$

As reported in [65], the worst-case error of the formulae as shown in (6.7)-(6.11) is 3.4% as compared to *EM solver* solutions. To get a better trade-off between accuracy and efficiency, *Matlab* calculations were conducted to check whether any multiple cases can be further approximated with a single formula. For different sets of symmetry parameters

(e.g., l, m or d), self inductances and mutual inductances for all 7 cases are calculated (i.e., L_{self} and M1-M7) using the formulae of (6.7)-(6.11), as reported in Table 6-1.

In Table 6-1, L_{self} refers to the self inductance and M1-M7 refers to the mutual inductances of cases 1-7. The self inductances for all the scenarios are 102.7, since this inductance is calculated for a base wire with fixed length (i.e., $l = 30$) and width (i.e., $w = 1$). The geometric parameters of l, m, and d, in the first three columns, refer to the length of a base wire, the length of the inducing wire and the separation in between these two wires. The N/As in Table 6-1 indicate that, some inductance formulae become invalid for certain scenarios. In the last four columns, we defined four fluctuation ratios called E1, E2, E3 and E4 that are calculated as shown in (6.12) - (6.15), respectively,

$$\mathbf{E1} = [max(\boldsymbol{M2}, \boldsymbol{M3}, \boldsymbol{M4}) - min(\boldsymbol{M2}, \boldsymbol{M3}, \boldsymbol{M4})]/L_{self}, \tag{6.12}$$

$$\mathbf{E2} = [max(\boldsymbol{M5}, \boldsymbol{M6}, \boldsymbol{M7}) - min(\boldsymbol{M5}, \boldsymbol{M6}, \boldsymbol{M7})]/L_{self}, \tag{6.13}$$

$$\mathbf{E3} = \max(|\boldsymbol{M3} - \boldsymbol{M2}|, |\boldsymbol{M3} - \boldsymbol{M4}|)/L_{self}, \tag{6.14}$$

$$\mathbf{E4} = \max(|\boldsymbol{M6} - \boldsymbol{M5}|, |\boldsymbol{M6} - \boldsymbol{M7}|)/L_{self}, \tag{6.15}$$

where $max(M2, M3, M4)$ and $min(M2, M3, M4)$ refer to the maximum and minimum among $M2$, $M3$ and $M4$, and $max(M5, M6, M7)$ and $min(M5, M6, M7)$ refer to the maximum and minimum among $M5$, $M6$ and $M7$. Thus, E1 reflects the fluctuation among $M2$, $M3$ and $M4$, and E2 reflects the fluctuation among $M5$, $M6$ and $M7$, with respect to self inductance L_{self}. Moreover, E3 reflects the worst-case ratio of the fluctuations between $M3$ & $M2$ and $M3$ & $M4$, and E4 reflects the worst-case ratio of the fluctuations between $M6$ & $M5$ and $M6$ & $M7$, with respect to self inductance L_{self}.

Table 6-1 Matlab statistics of mutual inductance calculations.

Geometries			Mutual Inductances (L_{self}= 102.7)						E1	E2	E3	E4
l	d	m	M2	M3	M4	M5	M6	M7	≤	≤	≤	≤
30	5	5	8.9	7.7	9.4	2.8	4.7	2.8	2%	2%	2%	3%
		10	17.7	15.9	N/A	5.6	8.7	5.6	2%	3%	2%	3%
		15	26.2	24.8	19.6	8.6	11.8	8.6	7%	2%	2%	3%
		20	34.1	33.6	29.5	11.8	14.3	11.8	5%	3%	1%	3%
		25	40.9	41.8	35.1	9.6	16.4	9.6	6%	7%	1%	7%
30	10	5	5.5	6.8	5.1	2.8	2.2	2.8	2%	1%	2%	1%
		10	10.8	11.5	N/A	5.6	6.2	5.6	2%	1%	1%	1%
		15	15.8	16.9	13.1	8.5	9.3	8.5	4%	1%	2%	1%
		20	20.2	22.3	17.5	11.8	11.8	11.8	5%	0%	3%	0%
		25	23.6	27.0	20.6	15.5	13.9	15.5	4%	2%	4%	3%
30	15	5	3.4	7.2	5.5	2.8	N/A	2.8	5%	0%	4%	0%
		10	6.7	9.9	N/A	5.6	1.24	5.6	4%	5%	4%	5%
		15	9.7	13.3	7.7	8.6	4.3	8.6	5%	5%	4%	5%
		20	12.1	16.7	9.6	11.8	6.8	11.8	7%	5%	5%	5%
		25	13.5	19.3	11.7	15.5	8.9	15.5	8%	7%	5%	6%
30	20	5	3.0	8.3	N/A	2.8	N/A	2.8	7%	0%	5%	0%
		10	3.8	9.5	N/A	5.6	1.24	5.6	6%	5%	6%	4%
		15	5.4	11.5	8.2	8.6	4.3	8.6	6%	5%	6%	4%
		20	6.4	13.4	12.6	11.8	6.8	11.8	7%	5%	6%	5%
		25	6.3	14.7	16.3	15.5	8.9	15.5	10%	7%	7%	7%

In Table 6-1, the dimensions of m and d (i.e., $m \in (l/6, 5l/6)$ and $d \in (l/6, 2l/3)$) are used because these conditions reflect the geometries of mutual inductances for our retargeting applications. For example, mutual inductances are calculated for parallel wires within a reasonable distance (i.e., $l \geq 2d$) in between. This is due to the following reasons: (1) mutual inductances for long-distance and short-length wires are negligible compared to their self-inductance counterparts because the magnetic flux linking the wires in such cases is very small; (2) when the distance becomes comparable to wire length, we have to

resort to expensive field-solver calculations because no analytical formulae thus far can give reasonable estimation of mutual inductances for such cases; (3) since interconnect wires in analog layouts are mostly long compared to the distance between wires, the considered cases (i.e., with the threshold of $l \geq 2d$) already cover all appreciable mutual inductances in our applications.

For all scenarios as shown in Table 6-1, E1 and E2 are both within 8% when $l \geq 2d$ (e.g., $d = \{5, 10, 15\}$) and within 10% when $l \leq 2d$ (e.g., $d = 20$). More important, E3 and E4 are both within 7% for all the scenarios. These observations shows that the difference among M2, M3 and M4 (with same geometries) is very small compared to self inductance, and so is the difference among M5, M6 and M7. These cases are not worth handling using distinct formulae considering the added solving complexity, especially for large-sized layouts that have huge number of interconnect wires. Thus, we approximate M2 and M4 by M3 (i.e., the simplest among these three), and approximate M5 and M7 by M6. As shown in Figure 6-2, cases 2 to 4 are approximated by case 3 (as enframed in the figure), and cases 5 to 7 are approximated by case 6. Therefore, mutual inductances for all wiring structures can be estimated by using three analytical formulae of M1, M3 and M6. Thus, geometric parameter of p, q and s in (6.7), (6.9) and (6.10) can be excluded from mutual inductance expressions, which greatly improves the solving efficiency.

6.3.2 Mutual Inductance Extraction

An appreciable mutual inductance arises when two parallel long wires with current flow are placed within limited horizontal and vertical distance, even if they do not overlap

with each other. Different from coupling capacitances that only consider overlapping parts of two wires as discussed in Section 6.4, in theory all geometries of parallel wires are required in the mutual-inductance formulae as shown in (6.7)-(6.11). Moreover, the type of a parallel-wire structure must be determined first to apply a correct formula.

The type of a parallel-wire structure is determined by the relative positions of its two member wires. As shown in Figure 6-3, a tile (either horizontal or vertical) deploys geometric coordinates for its bottom-left and top-right corners within a layout. For example, the coordinate of (x_{la}, y_{la}) represents the bottom-left corner of *Tile_a* in Figure 6-3. These parameters have fixed values in an extracted layout template to represent the layout geometries for tiles. These fixed values can be used to construct a set of scenarios of relative positions for parallel wires. As discussed in Section 6.3.1, the type of a wiring structure must be classified into one of the 7 cases. In order to detect the relative positions of a pair of wires, a set of logic conditions are designed as shown in Figure 6-4. It is to be noted that, the relative positions of layout tiles are preserved in retargeting as much as possible since the original layout is expertise-embedded. Thus, the mutual-inductance formulae for all scenarios (i.e., the relative-position assumptions) can always hold before, within and after a retargeting.

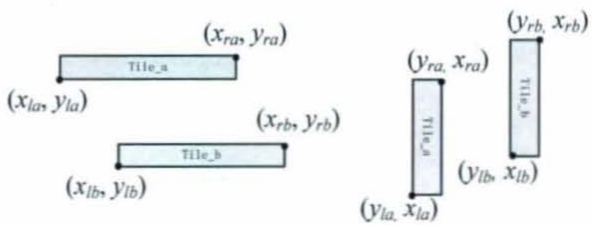


Figure 6-3 The geometric parameters of horizontal and vertical tiles.

In Figure 6-4, each pair of interconnect tiles, which is considered in mutual inductance calculation, is classified into a wiring type that corresponds to a unique mutual inductance formula. These logic conditions are shown in the lines of 2, 4, 6, 8, 10, 12 and 14 followed by their determined wiring types as well as mutual inductance formulae as shown in the lines of 3, 5, 7, 9, 11, 13 and 15, respectively. For example, in line 4, the logic condition is $(x_{ra} > x_{lb} \,\&\, x_{rb} > x_{la} \,\&\, (x_{rb} - x_{ra})(x_{lb} - x_{la}) > 0)$, and any pair of tiles that meets this condition should obviously be classified into case 4 and its mutual inductance will be approximated using M3. As a whole, any parallel wiring structure can be classified into one of the 7 cases, and its mutual inductance can be calculated using formula M1, M3 or M6.

Moreover, once the type of a wiring structure is determined, the parameters of l, m and d used by (6.5)-(6.11) must be represented with related tile geometries. Figure 6-5 shows an example of representing these parameters. After verifying the logic conditions as shown in Figure 6-4, this pair of wires is classified to case 4 and the applied formula is M3 as shown in (6.8). Here, the lengths of both wires and the vertical distance in between are represented with tile-geometry parameters (e.g., $d = (y_{la} - y_{rb})$). By applying these symbolic expressions in (6.8), the mutual inductance for this pair can be constructed in a layout-geometry form. Other cases of parallel wires rather than case 4 can be derived similarly. Thus, all the mutual inductances can be extracted and represented as geometric expressions. These expressions are then incorporated into the parasitic-aware RF formulation.

```
mExtraction
Begin
For any pair of parallel tiles named Tile_a and Tile_b Do
  if (x_la = x_ra & x_ra = x_rb)
      Wiring style belongs to case 1 using M1 for mutual inductance.
  else if (x_ra > x_lb & x_rb > x_la & (x_rb - x_ra)( x_lb - x_la) > 0)
      Wiring style belongs to case 4 using M3
  else if (x_ra > x_lb & x_rb > x_la & (x_la = x_lb || x_ra = x_rb))
      Wiring style belongs to case 3 using M3
  else if ((x_rb = x_ra)(x_la = x_lb) > 0))
      Wiring style belongs to case 2 using M3
  else if (x_lb > x_ra)
      Wiring style belongs to case 5 using M6
  else if (x_lb = x_ra)
      Wiring style belongs to case 6 using M6
  else
      Wiring style belongs to case 7 using M6
  end if
End for
End
```

Figure 6-4 An algorithm for the mutual inductance extraction.

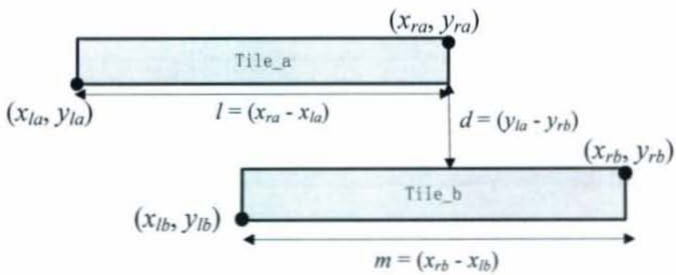


Figure 6-5 An example of geometric extraction for mutual inductance calculation.

6.4 Extraction of Wire-Coupling Capacitances

The wire-coupling capacitance C_{coup} arises when two parallel wires (e.g., interconnect tiles) overlap with each other. As shown in (5.3), coupling capacitance between two parallel tiles is calculated with the overlapping length and the perpendicular distance. It is to be noted that, an effective wire-coupling capacitor is only the overlapping part for a pair of wires. A pair of coupling wires can reside on the same layer as shown in Figure 6-6(a) or distinct layers as shown in Figure 6-6(b). The distance in Figure 6-6(b) can be calculated as:

$$dist = \sqrt{d\_ver^2 + d\_hor^2}\ , \tag{6.14}$$

where *d_ver* is the vertical distance between *layer* 1 and *layer* 2, *d_hor* is the horizontal distance between *tile* 1 and *tile* 2, and *dist* is the effective distance for the coupling capacitance.

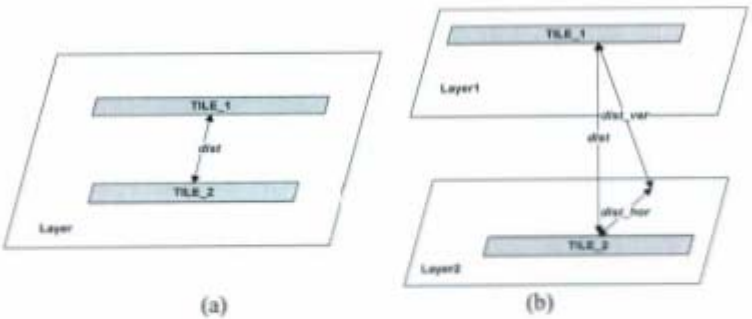


Figure 6-6 (a) Intra-layer and (b) inter-layer coupling capacitances.

One parasitic tile may overlap with another tile or other multiple tiles in complex wiring structures. For the extraction of these coupling capacitances, all tiles that overlap with a starting tile within certain distance can be detected using an algorithm called *overlapSearch*1 as shown in the pseudo-code of Figure 6-7. For a starting parasitic tile as shown in line 1, all the parasitic tiles that overlap with this starting tile are found for coupling capacitance calculation. This algorithm assumes that the impact of intermediate tiles is ignored in considering a pair of tiles that overlap with each other. For example, in Figure 6-8, the existence of *tile_2* is ignored (i.e., regarding this tile as an empty one) when searching *tile_4* from *tile_1*. In this example, more than 4 tiles are found to overlap with *tile_1* using *overlapSearch*1.

This algorithm has a complexity of $\mathbf{O}(n^2)$ where n is the total number of parasitic tiles for a layout. However, it is not correct to ignore the impact of intermediate tiles when searching a coupling wire. Moreover, considerable calculation effort is needed due to the excessive number of extracted coupling capacitances by *overlapSearch*1. As a matter of fact, the coupling capacitance for a pair is very small when other metal tiles are placed in between. These coupling cases are not worth handling in the RLC interconnect evaluation when considering both the correctness and the solving efficiency. Thus, the *overlapSearch*1 is an inferior algorithm for the coupling search.

To obtain correct and fast coupling search, the extraction of coupling capacitances is modified to an algorithm of *overlapSearch2* as illustrated in Figure 6-9. The impact of intermediate tiles is taken into account in this algorithm. For any parasitic tile, once an overlap is detected between the base tile and a found tile, the overlapping part is disabled in the next iteration of coupling search. As shown in the lines of 3-6, after the first

overlap is detected (i.e., between *tile_1* and *tile_2*), a new fake tile called *dummy_1* is defined as the remainder of *tile_1*. This dummy tile represents the part of the base tile that does not overlap with the found *tile_2*. The next iteration of searching is then conducted only for the tiles that overlap with *dummy_1*. The process ends when a base tile is completely covered.

```
    overlapSearch1
    Begin
1   For any starting tile (called tile_1) ∈ {parasitic tiles} Do
2      Search the nearest tile (called tile_2) to tile_1;
3      Extract the overlap between tile_1 & tile_2;
4      Search the second nearest tile to tile_1, say tile_3;
5      Extract the overlap between tile_1 & tile_3;
6      Repeat searching for all the parasitic tiles that overlap with tile_1;
7   End for
    End
```

Figure 6-7 Pseudo-code of *overlapSearch*1.

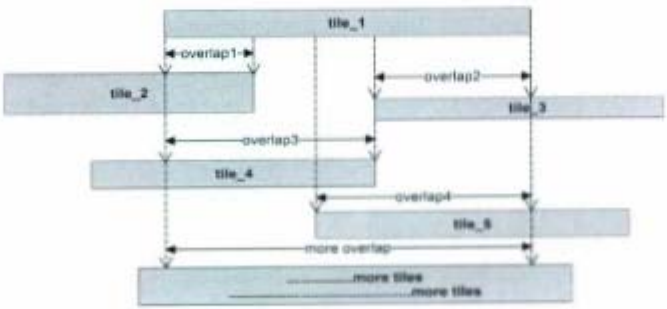


Figure 6-8 An example for *overlapSearch*1

```
overlapSearch2
Begin
For any starting tile (called tile_1) ∈ {parasitic tiles} Do
    Search the nearest tile (called tile_2) to tile_1;
    Extract the overlap between tile_1 & tile_2;
    Define a dummy tile called dummy_1 as the remainder of tile_1.
    Search the second nearest tile to tile_1, say tile_3;
    Extract the overlap between dummy_1 & tile_3;
    Define a dummy tile called dummy_2 as the remainder of tile_1.
    Repeat searching until tile_1 is completely overlapped;
End for
End
```

Figure 6-9 Pseudo-code of *overlapSearch2*

Figure 6-10 shows an example of the coupling searching using *overlapSearch2*. In this example, *tile_1* is the base tile and *tile_2* - *tile_4* are the tiles that overlap with *tile_1*. First of all, *tile_2* is detected as the nearest tile that overlaps with *tile_1* (i.e., *overlap1* in Figure 6-10). However, *overlap*1 does not completely cover *tile*_1. Then the remainder of *tile*_1 is defined as a virtual tile called *dummy*_1, which is defined as the current base tile to detect *tile*_3 and *overlap*2. Similarly, *tile*_4 and *overlap*3 are detected as the last tile to be found because *tile*_1 is now completely covered. With the same wiring structures as Figure 6-8, *overlapSearch2* found 3 tiles for coupling capacitances compared to over 4 tiles found by *overlapSearch1*.

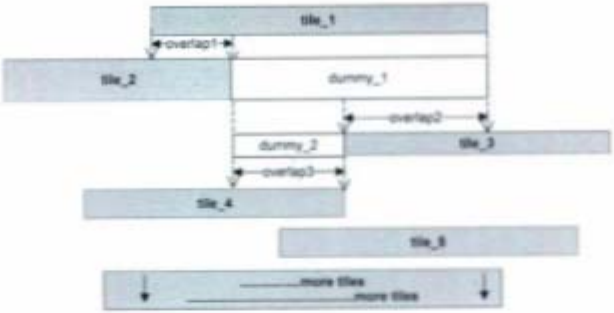


Figure 6-10 An example for *overlapSearch2*

The *overlapSearch2* has a worst-case complexity of $\mathbf{O}(n^2)$ where n is the total number of parasitic tiles of a layout. The complexity is reduced to $\mathbf{O}(n \times v)$ if the worst case number of overlapping parasitic tiles for a base tile is v, since ($v < n$) always holds for a layout. As a whole, the *overlapSearch2* achieves efficient coupling search by only taking into account correct coupling capacitances between the tiles with no intermediate impacts.

6.5 Retargeting Algorithm Design

In order to ensure the desired RF circuit performance, the performance deviation due to interconnect RLCs must be restricted within a maximum allowed tolerance. Therefore, the RF performance constraints can be represented as:

$$\sum S_{res} \times R\ (x,y) + \sum S_{sub} \times C_{sub}(x,y) + \sum S_{coup} \times C_{coup}(x,y) + \sum S_{ind} \times L(x,y) \le \Delta W_{max}, \quad (6.15)$$

where $R(x, y)$, $C_{sub}(x, y)$, $C_{comp}(x, y)$ and $L(x, y)$ refer to geometric expressions of parasitic resistance, wire-substrate capacitance, wire-coupling capacitance and parasitic inductance, respectively. S_{res}, S_{sub}, S_{comp}, and S_{ind} represent the performance sensitivities with respect to parasitic resistances, wire-substrate capacitances, wire-coupling capacitance and parasitic inductances, respectively.

With the RF performance constraints as shown in (6.15), the parasitic-aware RF layout retargeting can be formulated as a two-dimensional MINLP problem as specified in (6.16)-(6.20), besides basic constraints for design rules and symmetry. In this formulation, geometric expressions of parasitic inductances and wire-coupling capacitances are integrated as shown in (6.17) to enable the capability of RF layout retargeting. This formulation is solved following the schemes specified in Section 5.5.2.

$$\textit{Minimize} \quad (x_{rr} - x_{ll}) \times (y_{rr} - y_{ll}), \tag{6.16}$$

Subject to

$$\sum S_{res} \times R\ (x, y) + \sum S_{sub} \times C_{sub}(x, y) + \sum S_{comp} \times C_{comp}(x, y) + \sum S_{ind} \times L(x, y) \le \Delta W_{max} \tag{6.17}$$

$$P_{m1}(x, y) = P_{m2}(x, y), \tag{6.18}$$

$$\sum_{n=1}^{N_{res}} B_n = \sum_{n=1}^{N_{sub}} B_n = \sum_{n=1}^{N_{ind}} B_n = 1,\ B_n \in \{0, 1\}, \tag{6.19}$$

$$B_n\,(P - P_{n1}) \ge 0\ \&\&\ B_n\,(P - P_{n2}) \le 0, \tag{6.20}$$

where x_{rr}, x_{ll}, y_{rr} and y_{ll} represent the boundaries of the entire RF layout.

6.6 Experimental Results

The proposed RF retargeting algorithm is integrated into IPRAIL. In this section, we report retargeting results on three analog and RF circuits: a *two-stage Miller-compensated operational amplifier* as shown in Figure 4-4, *a single-ended folded cascode opamp* as shown in Figure 4-5 and *a double-ended low-noise amplifier* (*LNA*) as depicted in Figure 6-11. The LNA (operating at 5.6 GHz) was retargeted to meet new specifications in a 0.18μm MITLL SOI technology process.

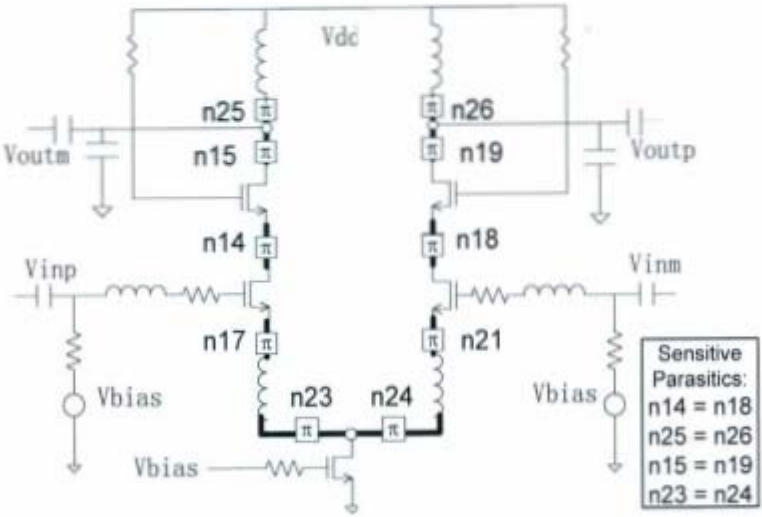


Figure 6-11 A double-ended LNA.

First, parasitic-aware retargeting was conducted on three analog/RF designs to verify the superior effectiveness and efficiency of *overlapSearch*2 compared to *overlapSearch*1 for wire-coupling extraction. Table 6-2 reports the number of extracted wire-coupling pairs and parasitic solving times for both algorithms, and Table 6-3 reports the results of post-layout simulations. For example, in Table 6-2, the number of extracted coupling pairs for the *LNA* is 16 by using *overlapSearch*1 compared to 8 by using *overlapSearch*2. Moreover, the parasitic solving time for the two-stage opamp by using *overlapSearch*1 is 64.0s compared to 40.3s by using *overlapSearch*2. In Table 6-3, retargeting by using both algorithms meets the performance specification for the opamps, but *overlapSearch*1 renders the RF retargeting for the LNA infeasible while *overlapSearch*2 works. From the above observations, *overlapSearch*2 produced much fewer extracted wire-coupling pairs (e.g., 50% fewer for the LNA) and significant reduction of parasitic solving time compared to *overlapSearch*1. In particular, *overlapSearch*2 realized a 33% reduction of parasitic solving time for the two-stage and cascode opamps, whereas *overlapSearch*1 even rendered the problem unsolvable for the LNA due to adding wrongly extracted wire-coupling capacitances. Therefore, *overlapSearch*2 is adopted in the RF retargeting flow.

Table 6-2 Number of extracted wire-coupling pairs and parasitic solving times by *overlapSearch*1 and *overlapSearch*2.

	Number of Extracted Pairs		**IPOPT Solving Time**	
	***overlapSearch*1**	***overlapSearch*2**	***overlapSearch*1**	***overlapSearch*2**
Two-Stage	34	24	64.0s	40.3s
Cascode	37	31	137.5s	99.2s
LNA	16	8	unsolvable	126.8s

Table 6-3 Post-layout simulations for *overlapSearch*1 and *overlapSearch*2.

		Gain (dB)	BW (MHz)	PM (°)	GM (dB)	Area (μm^2)
Two-stage opamp	***Specification***	60.0	100	90.0	10.0	-
	overlapSearch1	64.3	103.4	90.5	16.6	2815
	overlapSearch2	64.3	106.4	90.5	16.7	2815
Folded-cascode opamp	***Specification***	60.0	60.0	60.0	10.0	-
	overlapSearch1	60.6	61.7	60.5	10.4	2201
	overlapSearch2	60.6	63.7	61.1	10.4	2190
		S11 (dB)	**NoiseFigure (dB)**	**Gain (dB)**	**IIP3 (dB)**	**Area (μm^2)**
LNA	***Specification***	<-15.0	<2.0	>10.0	>-9.0	-
	overlapSearch1	Completely failed				
	overlapSearch2	-20.20	1.08	15.61	-8.76	0.618

The performance sensitivities with respect to wire-coupling capacitances are reported in Table 6-4 for the opamps. For example, in the second row, the *C2d_2e* refers to the coupling capacitances between the nets of *n2d* and *n2e*, and the sensitivity of *bandwidth* with respect to *C2d_2e* is -0.105 MHz/fF. To demonstrate the superior effectiveness of the proposed layout retargeting with wire-coupling capacitances considered, a retargeting flow called PM-RCC was setup. PM-RCC considers parasitic resistances, wire-substrate capacitances and wire-coupling capacitances in the retargeting. Layout retargeting was then conducted on the two opamps using PM-RCC as well as PM-RC (i.e., only considers parasitic resistances and wire-substrate capacitances).

The post-layout simulations for PM-RC and PM-RCC were conducted and the results are reported in Table 6-5. The observation is that, both methods achieve satisfactory circuit performance while PM-RCC obtains a moderate improvement of

bandwidth as well as layout area. In particular, the achieved bandwidth by PM-RCC is 107.0 MHz compared to 106.4 MHz by PM-RC, and achieved layout area is 2190 μm^2 by PM-RCC compared to 2201 μm^2 by PM-RC. Thus, the consideration of wire-coupling capacitances can improve the quality of target layouts even for lower-frequency analog designs.

Table 6-4 Performance sensitivities with respect to wire-coupling capacitances.

Design	Ccoup	Performance Sensitivities of.			
		Gain	Bandwidth	PhaseMargin	GainMargin
2STAGE	C2d_2e	0	-0.105	0.003	-0.072
	C2c_3c	0	-0.095	0.009	0.045
	C3a_3b	-0.13	0.021	-0.001	-0.091
CASCODE	C1a_4a	0	0.00216	-0.034	-0.019
	C1b_4b	-0.001	0.00216	-0.034	-0.005
	C1c_4c	0	0.00216	-0.034	-0.056

Table 6-5 Post-layout simulations for target layouts by PM-RC/PM-RCC.

		Gain (dB)	BW (MHz)	PM (°)	GM (dB)	Area (μm^2)
Two-stage opamp	Spec.	60.0	100	90.0	10.0	-
	PM-RC	64.3	106.4	90.5	16.7	2815
	PM-RCC	64.3	107.0	90.5	16.7	2815
Folded-cascode opamp	Spec.	60.0	60.0	60.0	10.0	-
	PM-RC	60.6	63.7	61.1	10.4	2201
	PM-RCC	60.6	63.7	61.1	10.4	2190

Before the RF retargeting on the LNA, performance sensitivities and upper bounds for parasitic resistances, wire-coupling capacitances and inductances were generated through simulations. These simulations with variable parasitics were conducted using *Cadence Ocean Script* on the double-ended LNA operating at 5.6 GHz. The lumped RLC interconnect model as shown in Figure 6-1 is applied in modeling RF interconnects. RF performances of *S11*, *Noise Figure*, *Power Gain* and *IIP3* were considered in retargeting.

Here, the number of segments for sensitive parasitic inductance is determined based on the criterion as shown in Table 6-6, besides the criterion for parasitic resistances and capacitances as discussed in chapter 5. Piecewise performance sensitivities with respect to parasitic resistances, inductances and wire-coupling capacitances are reported in Table 6-7. For example, the inductances of a matching parasitic *L25&L26* have an upper bound of *0.0069* nH and a piecewise sensitivity of $(2.5B_1+4.1B_2)$ dB/nH with respect to *power gain* as shown in Table 6-7.

Table 6-6 Criterion of determining the number of segments for parasitic inductances.

$\lvert S_{i\_max}\rvert$	$\lvert\alpha\rvert$	N_{seg}
≤ 1.0	Any	1
Any	> 0.5	1
[1.0, 10.0]	(0, 0.5]	2
(10.0, 15.0]	[0.3, 0.5]	3
	(0.1, 0.3)	4
> 15.0	Any	4
Any	(0, 0.1]	4

Table 6-7 Performance sensitivities with respect to parasitic inductances, resistances and wire-coupling capacitances for the LNA.

RLC (Ω, nH, fF)	Sens-S11	Sens-NF	Sens-Gain	Sens.-IIP3
L25=L26	$-1.1B_1+2.14B_2$	$-1.5B_1-1.8B_2$	$2.5B_1+4.1B_2$	0.17
L14=L18	$-5.0B_3-8.1B_4$	-0.021	0.69	-0.14
L15=L19	-0.2	0.009	$0.4B_5+0.55B_6$	0.12
L23=L24	$2.1B_7+3.3B_8$	$-0.21B_7+0.31B_8$	$-1.5B_7-2.9B_8$	-0.09
R25=R26	0.18	0.13	$-0.76B_1-0.922B_2$	0.07
R14=R18	0.067	0.031	-0.29	0.024
R15=R19	-0.18	0.11	$-0.86B_5-1.05B_6$	-0.047
R23=R24	$0.31B_7+0.57B_8$	0.25	$-0.93B_7-2.1B_8$	-0.011
C14_18	$0.05B_3-0.08B_4$	0	0	-0.007
C15_19	-0.02	0.001	0	0.022
C23_24	0.01	0.001	0	-0.005

Then we conducted RF retargeting on the LNA using the proposed RF retargeting method (called PM-RLC). To demonstrate the superior effectiveness of RLC retargeting compared to traditional retargeting (i.e., only considering resistive parasitics) for an RF layout, a similar flow called PM-R was setup. The PM-R follows the PM-RLC retargeting flow but excludes parasitic inductive and wire-coupling capacitive impacts. Both methods were used for retargeting the LNA and their results were compared. The post-layout simulations were conducted for the target layouts generated by PM-R and PM-RLC. The simulated performances are summarized in Table 6-8.

A key observation of Table 6-7 is that, noise-figure sensitivities with respect to all parasitic resistances are positive (i.e., degrading/increasing the noise figure due to increasing parasitic resistances), whereas some counterparts of inductance sensitivities are

negative (i.e., improving/reducing the noise figure due to increasing parasitic inductances). This actually accounts for the obvious noise figure improvement of PM-RLC over PM-R (i.e., 1.08 < 1.82) in Table 6-8. A similar performance improvement for the power gain is achieved by PM-RLC due to large positive sensitivities with respect to inductances whereas all the counterpart sensitivities with respect to resistances are negative. Thus, PM-RLC optimizes contributing parasitic inductances so that it achieves the best performances and the smallest layout area. The original and target layouts of the LNA by PM-RLC are depicted in Figure 6-12. Longer and obviously thinner interconnect wires can be observed in the target layout. The longer and thinner interconnects in the target layout result in larger parasitic inductances for *L25* & *L26* and *L14* & *L18*, which improves *noise figure* and *power gain*, respectively.

Table 6-8 Post-layout simulations for target LNA layouts by PM-R/PM-RLC.

	S11 (dB)	NoiseFigure (dB)	Gain (dB)	IIP3 (dB)	Area (μm^2)
Specification	<-15.0	<2.0	>10.0	>-9.0	-
PM-R	-20.11	1.82	11.95	-8.76	0.630
PM-RLC	-20.20	1.08	15.61	-8.76	0.618

Execution time is reported in Table 6-9 for RF retargeting using PM-R and PM-RLC. PM-RLC effectively manages the RF retargeting within 6 minutes of CPU time. However, the execution time of PM-RLC is 45% more than that of PM-R due to the added complex inductance and wire-coupling capacitance formulae.

Table 6-9 Time efficiency of the RF retargeting using a very tight error tolerance for IPOPT.

	LNA (s)	
	PM-R	**PM-RLC**
Template Extraction	146	146
Layout Generation	230	325
Parasitic Solving	36.5	126.8

To reduce the solving time for PM-RLC, an updated threshold of error tolerance for IPOPT must be applied to shrink the execution time of PM-RLC. This threshold defines the solving effort of IPOPT. Table 6-10 reports the realized objective functions and parasitic solving times for different thresholds of error tolerances. The first column lists the thresholds ranging from 1×10^{-8} to 5×10^{-6}, and the middle two columns record the achieved objective functions by PM-R/PM-RLC. The last two columns report the IPOPT solving times for both methods when applying different thresholds. In our RF retargeting, the layout area is designed as the objective function and this function should be minimized by IPOPT in an optimization process. It is worth mentioning that, optimal solutions can be found by IPOPT only when the error tolerance is reduced below its threshold. As shown in Table 6-10, the best objective function and longest execution time are caused by using a very tight threshold of 1×10^{-8}, while worse objective functions and shorter execution times are caused by using loosened thresholds. To reduce the parasitic solving time of PM-RLC without compromising the quality of solutions, a trade-off threshold of 1×10^{-6} was adopted for the RF retargeting. As shown in Table 6-10, very similar objective function is achieved using a threshold of 1×10^{-6} compared to 1×10^{-8}, but

the execution time by using the trade-off threshold is two times less than that of the tight threshold. By using the tradeoff threshold, the execution time of PM-RLC retargeting as greatly reduced without sacrificing the quality of layout solutions.

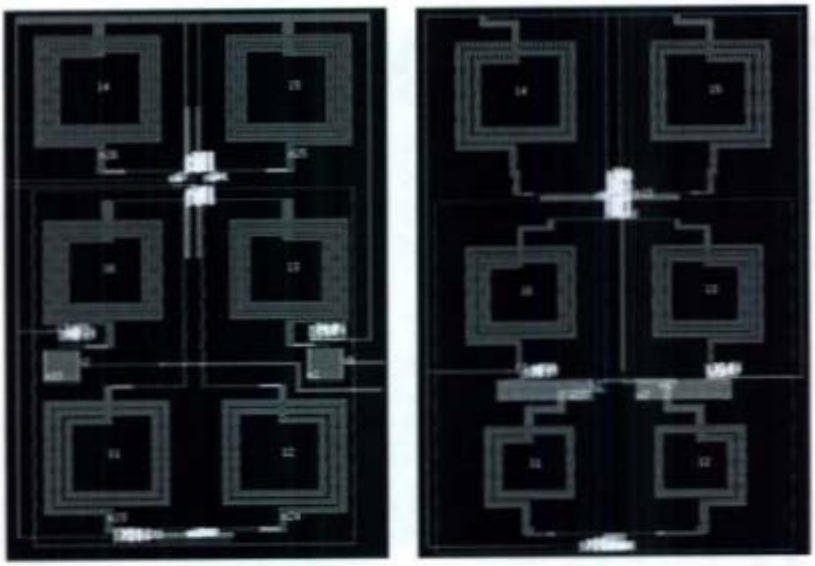
The updated time efficiency is reported in Table 6-11, where no appreciable time increase can be found for PM-RLC compared to PM-R. With similar execution time, PM-RLC achieves improved RF performance and reduced layout area for the LNA.

Table 6-10 Time efficiency of the RF retargeting using different thresholds of error tolerance for IPOPT.

IPOPT Tolerance	Objective Function		Solving Time (s)	
	PM-R	PM-RLC	PM-R	PM-RLC
1×10^{-8}	22984937	22984937	36.5	126.8
5×10^{-8}	22984937	22984938	36.1	109.5
1×10^{-7}	22984937	22984938	35.7	63.7
5×10^{-7}	22984937	22984938	34.4	35.3
1×10^{-6}	22984937	22984938	31.5	33.2
5×10^{-6}	22993174	23132806	13.1	21.5

Table 6-11 Time efficiency of the RF retargeting after applying a loosened error tolerance for IPOPT.

IPOPT Tolerance $= 1\times10^{-6}$	LNA (s)	
	PM-R	PM-RLC
Template Extraction	146	146
Layout Generation	227	231
Parasitic Solving	31.5	33.2



(a) (b)

Figure 6-12 (a) Original and (b) target layouts of the LNA by PM-RLC.

6.7 Summary

In this chapter, a performance-constrained parasitic-aware RF layout retargeting algorithm was presented. Different from previous methods, parasitic inductances and wire-coupling capacitances were incorporated into the parasitic-aware retargeting formulation, which enables an RF capability of the proposed approach. The experimental results show the proposed algorithm achieves effective retargeting of RF layouts, which achieves updated performance specifications as well as smaller layout area compared to its alternatives.

7. Conclusions and Future Work

7.1 Conclusions

Recently, analog designers still have to spend disproportionate time and effort in conducting handcrafted analog layout design, due to insufficient support from available analog CAD tools. This manual design style leads to a tedious and error-prone layout process in order to ensure a good tradeoff among design aspects, such as layout area minimization, performance optimization, noise reduction, power minimization, etc. The layout-geometry (e.g., symmetry, matching and parasitics) induced performance degradation challenges the automation of analog/RF layout design. Several current attempts for analog/RF layout automation fall into two general patterns: macrocell-based physical synthesis and template-based layout compaction.

An improved template-based analog/RF layout automation algorithm was presented in this dissertation for parasitic-aware process migration and/or performance update [7] [8]. The proposed performance-constrained retargeting flow was integrated into an automated layout tool called IPRAIL [6]. The implemented IPRAIL efficiently conducts effective layout generation for process migration and/or performance retargeting on several analog/RF layouts (each layout contains thousands of rectangular geometries). The layout retargeting was successfully managed within 3 minutes for the analog designs (i.e., both operational amplifiers) and 6 minutes for the RF design (i.e., the double-ended LNA).

Various innovative techniques were proposed for analog/RF layout automation, including piecewise sensitivity computation, mutual inductance modeling and extraction, coupling capacitance extraction, and the parasitic-aware retargeting formulation using mixed-integer nonlinear programming. Moreover, to make the proposed methodology capable of handling RF retargeting, a lumped RLC interconnect model was applied in RF interconnect solving. Global and accurate parasitic control were achieved by directly constraining performance degradations restricted within maximum allowed deviation with the aid of piecewise sensitivities. Analytical equations of mutual inductances and wire-coupling capacitances were employed in RF retargeting with reasonable accuracy.

Piecewise sensitivities are more accurate compared to single numeric performance sensitivities. For sensitive nets, the sensitivities are large and may vary significantly along with changing parasitics. Using single sensitivities, the generated parasitic solutions are usually outside of the domain where these single sensitivities are calculated. In the piecewise formulation, different segments of a parasitic are always matched with their segmental performance sensitivities so that the optimized parasitic solutions always fall into their applicable segments. To ensure a good trade-off between accuracy and efficiency, the number of segments is determined according to the proposed criterion tables. These piecewise models are then used to construct direct performance constraints.

Performance constraints are modeled as the performance deviations due to all interconnect resistances, self and mutual inductances, wire-substrate capacitances, and wire-coupling capacitance. The parasitic expressions are limited by their piecewise sensitivities (i.e., their contributions to performances) in the performance constraints. By incorporating all RLC-interconnect impacts into the performance constraints, the

correlation and cancellation among all parasitics are achieved in a global optimization towards satisfactory performance of target layouts.

The capability of RF-layout retargeting is another important contribution of this dissertation [5]. A lumped RLC interconnect model is designed to incorporate inductive impacts into the performance optimization besides resistive/capacitive impacts. An equation of self inductance for that model is represented with its related geometric parameters. For better accuracy of representing inductive impacts, mutual inductances for complex wiring structures are analyzed and then simplified while still accurate equations are applied in the interconnect calculations. Moreover, wire-coupling capacitances are also included for RF formulation since they become very active in affecting circuit performance at radio frequency. By applying the RLC interconnect model as well as a set of RLC equations into the proposed analog formulation, an RF retargeting formulation was constructed and successfully solved.

Moreover, mixed-integer nonlinear programming was applied as an effective solving technique for enforcing logic conditions within the retargeting formulation. It is inefficient to solve the complete constraint set which usually contains a very small portion of MINLP performance constraints (i.e., the number of these constraints is actually just the number of performance goals). Thus, a two-phase solving scheme was applied in the layout generation process and its efficacy is demonstrated in our experiments. By using this scheme, the 4-parameter symmetry/matching constraints as well as performance constraints are solved in a separate MINLP-only optimization phase. And the complete constraint set can be solved in the second LP-only phase.

Traditional analog/RF layout design methods typically require tedious iterations of simulations and handcrafted re-designs to meet updated performance specifications and/or updated process technologies. As an improvement, the parasitic-aware performance-constrained retargeting method, proposed in this dissertation, considerably improves the efficiency and effectiveness of analog/RF layout automation. As a whole, the IPRAIL updated with the proposed template-based methodology enables analog/RF designers to generate high-performance minimum-area layouts within minutes of CPU time.

7.2 Future Work

IPRAIL, with the proposed algorithms incorporated, handles analog/RF retargeting for building-block layouts such as multi-stage operational amplifiers. As future work, further development to IPRAIL can be conducted towards retargeting larger layouts including hierarchically organized multiple building-blocks. For larger sized analog/RF layouts, the extraction of layout template would be more complicated and the layout solving with hundreds of thousands of constraints are more challenging. The proposed algorithms can be extended to handle multi-block layouts if hierarchical decomposition algorithms and specification transmission techniques are available.

Figure 7-1 shows an example multi-block layout with hierarchical structures. A multi-block layout is first analyzed and decomposed into building blocks (i.e., functional units). For example, the layout in Figure 7-1 is decomposed into 4 building blocks. The performance specification for the layout can then be translated into the performance

requirements for each building block. As shown in Figure 7-1, the building blocks of block1, block2, block3 and block4 have their own specifications of spec1, spec2, spec3 and spec4, respectively. For each building block, the implemented IPRAIL in this dissertation can conduct automatic retargeting to generate a target building block. With all the target building blocks, a template-base parasitic-aware rebuild can be conducted to connect all the target building blocks and form a target layout.

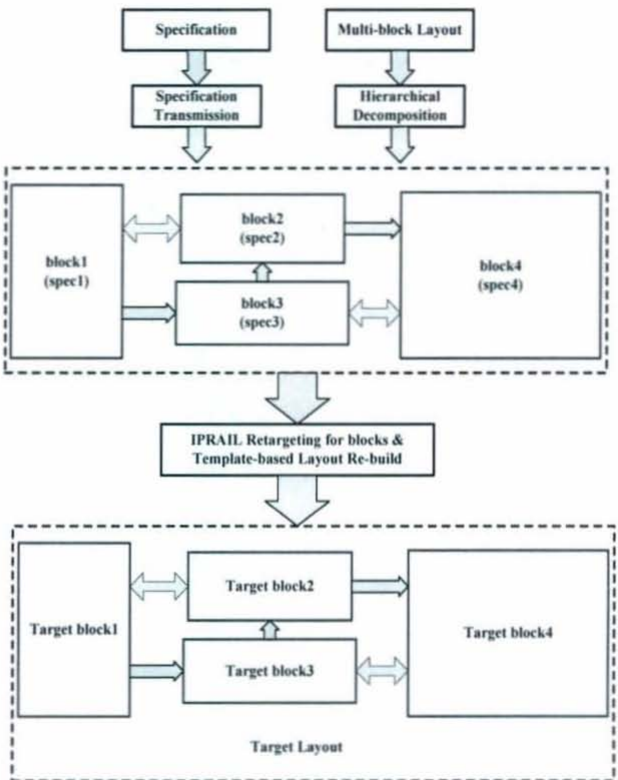


Figure 7-1 Prospect IPRIAL retargeting flow for multi-block layouts.

Moreover, customized solving schemes can be developed to solve larger-sized layout formulation, e.g., using the *C++* interfaces of IPOPT or other standard nonlinear solvers. The configuration of thresholds is very important for the process of IPOPT solving. For example, coding can be conducted to interact with IPOPT through a *C++* interface to achieve dynamic control of the solving process.

8. References

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

[2] M. J. Krasnicki, R. Phelps, J. R. Hellums, M. McClung, R. A. Rutenbar and L. R. Carley, "ASF: A practical simulation-based methodology for the synthesis of custom analog circuits," *Proc. IEEE/ACM International Conference on Computer-Aided-Design*, pp. 350-357, Nov. 2001.

[3] Arsyn User's Manual, *Orora Design Technologies Inc.*, 2003.

[4] N. Jangkrajarng, S. Bhattacharya, R. Hartono, and C. Shi, "IPRAIL—Intellectual property reuse-based analog IC layout automation," *Integration. VLSI J.*, vol. 36, no. 4, pp. 237–262, Nov. 2003.

[5] Zheng Liu and Lihong Zhang, "Performance-constrained template-driven retargeting for analog and RF layouts," *in Proc. IEEE/ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 429-434, 2010.

[6] Zheng Liu and Lihong Zhang, "A performance-constrained template-based layout retargeting algorithm for analog integrated circuits," *in Proc. Asia and South Pacific Design Automation Conference (ASPDAC)*, pp. 293 - 298, 2010.

[7] Zheng Liu and Lihong Zhang, "Performance-constrained parasitic-aware retargeting and optimization of analog layouts," *Proceedings of IEEE Canadian Conference on*

Electrical and Computer Engineering (*CCECE*), St. John's, Newfoundland, May 2009.

[8] Zheng Liu and Lihong Zhang, "Optimization of Parasitic Constraints for Analog Integrated Circuits," *Proceedings of Newfoundland Electrical and Computer Engineering Conference* (*NECEC*), St. John's, Newfoundland, Nov. 2008.

[9] L. Zhang, N. Jangkrajarng, S. Bhattacharya, and C. Shi, "Parasitic-aware optimization and retargeting of analog layouts: a symbolic-template approach," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 5, pp. 791–829, May. 2008.

[10] HSPICE ® User Guide, *Synopsys*, Nov 2005.

[11] HSPICERF ® User Guide, *Synopsys*, June 2006.

[12] DesignWare® Analog IP Quick Reference Guide, *Synopsys*, Sept 2010.

[13] IC Station Layout Suite Data Sheet, *Mentor Graphics Inc*, 2009.

[14] Laker-Custom Layout System Reference Guide, *SpringSoft Inc*, 2008.

[15] NeoCell® User's Guide, Version 3.3, 2003.

[16] NeoCircuit® User's Guide, Version 3.4, 2007.

[17] H. Y. Koh, C. H. Squin, and P. R. Gray, "OPASYN: A compiler for CMOS operational amplifiers," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 113-126, Feb. 1990.

[18] R. Harjani, R. A. Rutenbar, and L. R. Carley, "OASYS. A framework for analog circuit synthesis," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 1247-1266, Feb. 1989.

[19] H. Yaghutiel, A. Sangiovanni-Vincentelli, and P. R. Gray, "A methodology for automated layout of switched-capacitor filters," *in Proc. IEEE ICCAD*, pp. 444-447, 1986.

[20] G. Jusuf, P. R. Gray, and A. Sangiovanni-Vincentelli, "CADICS-Cyclic analog-to-digital converter synthesis," *in Proc. IEEE ICCAD*, pp. 286-289, Nov. 1990.

[21] J. D. Conway and G. Schrooten, "An automatic layout generator for analog circuits," *in Proc. Euro. Des. Autom. Conf.*, pp. 513–519, 1992.

[22] V. Meyer zu Bexten, "ALSYN: Flexible rule-based layout synthesis for analog IC's," *IEEE J. Solid-State Circuits*, vol. 28, no. 3, pp. 261–268, Mar. 1993.

[23] J. Rijmenants, J. B. Litsios, T. R. Schwarz, and M. G. R. Degrauwe, "ILAC: An automated layout tool for analog CMOS circuits," *IEEE J. Solid-State Circuits*, vol. 24, no. 4, pp. 417–425, Apr. 1989.

[24] E. Malavasi, E. Charbon, E. Felt, and A. Sangiovanni-Vincentelli, "Automation of IC layout with analog constraints," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 8, pp. 923–942, Aug. 1996.

[25] M. Cohn, D. J. Garrod, R. A. Rutenbar, and L. R. Carley, "KOAN/ANAGRAM II: New tools for device-level analog placement and routing," *IEEE J. Solid-State Circuits*, vol. 26, pp. 330-342, Mar. 1991.

[26] K. Lampaert, G. Gielen, and W. Sansen, *Analog Layout Generation for Performance and Manufacturability*, Boston: Kluwer Academic Publishers, 1999.

[27] L. Zhang, U. Kleine, and Y. Jiang, "An automated design tool for analog layouts," *IEEE Trans. on Very Large Scale Integration Systems*, vol. 14, pp. 881-894, Aug. 2006.

[28] M. J. M. Pelgrom, A. C. J. Duinmaijer and A. P. G. Welbers, "Matching Properties of MOS Transistors," *IEEE Journal of Solid State Circuits*, vol. 24, pp. 1433-1440, October 1989.

[29] A. Hastings, *The Art of Analog Layout*, Prentice Hall Incorporate, 2001.

[30] R. Castro-López, O. Guerra, E. Roca, and F. V. Fernández, "An Integrated Layout-Synthesis Approach for Analog ICs," *IEEE Trans. Computer-Aided-Design*, vol. 27, pp. 1179-1189, July. 2008.

[31] E. Yilmaz and G. Dündar, "Analog Layout Generator for CMOS Circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 1, pp. 32–44, Jan. 2009.

[32] D.J. Allstot, K. Choi and J. Park, *Parasitic-aware Optimization of CMOS RF Circuits*, Boston: Kluwer Academic Publishers, 2003.

[33] Awwad F. R., Lammoshi T., Nekili M., "Importance of on-chip inductance in designing RLC VLSI interconnects," *in Microelectronics, Microelectronics, the 14th International Conference (ICM). 2002.*

[34] M. Conti, P. Crippa, S. Orcioni, and C. Turchetti, "Layout-based statistical modeling for the prediction of the matching properties of MOS transistors," *IEEE Trans. Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, pp. 680-685, May 2002.

[35] B. Razavi, Design of Analog CMOS Integrated Circuits, McGraw Hill, 2001.

[36] Shah. S and Nunez. A, "Pre-layout parasitic estimation in interconnects," *Proc. 48th Midwest Symposium on Circuits and Systems, (MWSCAS)*, vol. 2, pp. 1442-1445, 2005.

[37] Sanga. S. and Yamau. S, "Post-layout parasitic verification methodology for mixed-signal designs using fast-SPICE simulators," *Proc. 4th IEEE Dallas Circuits and Systems Workshop on System-on-Chip (DCAS)*, pp. 211-214, *2005*.

[38] J. Lienig, G. Jerke and T. Adler, "Electro-migration avoidance in analog circuits: Two methods for current driven routing," *Proc. Asia and South Pacific Design Automation Conference*, pp. 372-378, Jan. 2002.

[39] M. Albina and G. Hackl, "Layout Parasitic Interconnections Effects on High Frequency Circuits," *Proc. 6th IEEE Dallas Circuits and Systems Workshop on System-on-Chip (DCAS)*, pp. 1-4, *2007*.

[40] P. G. Drennan and C. McAndrew, "Understanding MOSFET mismatch for analog design," *IEEE J. Solid-State Circuits*, vol. 38, no. 3, pp. 450–456, Mar. 2003.

[41] S. Bhattacharya, N. Jangkrajarng, R. Hartono, and C. Shi, "Hierarchical extraction and verification of symmetry constraints for analog layout automation," *Proc. Asia and South Pacific Design Automation Conference (ASPDAC)*, pp. 400 - 405, 2004.

[42] S. Bhattacharya, N. Jangkrajarng, and C. Shi, "Multilevel Symmetry-Constraint generation for Retargeting Large Analog Layouts," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System*, vol. 25, no. 6, pp. 945–960, May. 2006.

[43] Y. Tam, E. Young and C. Chu, "Analog Placement with Symmetry and Other Placement Constraints," *in Proc. IEEE ICCAD*, pp. 349-354, Nov. 2006.

[44] R. Castro-Lopez, F. V. Fernandez, F. Medeiro, and A. Rodriguez-Vazquez, "Generation of technology-independent retargetable analog blocks," *International*

Journal of Analog Integrated Circuits and Signal Processing, vol. 33, pp. 157-170, Dec. 2002.

[45] S. Bhattacharya, "Template-Driven Parasitic-Aware Optimization of Analog/RF IC Layouts," *PhD dissertation*, University of Washington, 2005.

[46] S. Rubin, *Computer Aids for VLSI Design*, Appendix B, Addison-Wesley, 1987.

[47] S. Bhattacharya, N. Jangkrajarng, R. Hartono and C.-J. R. Shi, "Correct-by-construction layout-centric retargeting of large analog designs," *Proc. IEEE/ACM Design Automation Conf.*, pp. 139-144, Jun. 2004.

[48] K.-W. Chiang, "Resistance extraction and resistance calculation in GOALIE2," *Proc. IEEE/ACM Design Automation Conf.*, pp. 682-685, Jun. 1989.

[49] A. Wachter and L. T. Biegler, "On the implementation of an interior point filter line-search algorithm for large-scale nonlinear programming," *Math.Program*, vol. 106, no. 1, pp. 25–57, Apr. 2005.

[50] Cadence Analog Design Environment User Guide, Product Version 5.0, CADENCE, April 2004.

[51] Virtuoso Spectre Circuit Simulator User Guide, Product Version 7.0.1, June 2008.

[52] U. Choudhury, "Sensitivity Computation in SPICE3," *Masters Dissertation*, U.C. Berkeley, Dee: 1988.

[53] OCEAN Reference, Product Version 5.1.4.1, June 2004.

[54] The MOSEK Optimization Tools Manual, version 5.0, MOSEK, 2008.

[55] LINGO User's Guide, version 9.0, LINDO Systems Inc, 2008.

[56] U. Choudhury and A. Sangiovanni-Vincentelli, "Use of performance sensitivities in routing analog circuits," *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 348-351, May 1990.

[57] U. Chouduy, A. Sangiovanni Vincentelli, "Automatic Generation of Parasitic Constraints for Performance Constrained Physical Design of Analog Circuits," *IEEE Trans. on CAD of Integrated Circ. and Syst.*, vol.12, pp.208-224, Feb. 1993.

[58] U. Choudhury and A. Sangiovanni-Vicentelli, "Constraint Generation for Routing Analog Circuits," *27th ACM/IEEE Design Automation Conference*, pp. 561-566, 1990.

[59] T. Sakurai, "Closed-form expressions for interconnection delay, coupling, and crosstalk in VLSI's," *IEEE Trans. Electron Device*, vol. 40, pp. 118-124, Jan. 1993.

[60] R. Okuda, T. Sato, H. Onodera, and K. Tamaru, "An efficient algorithm for layout compaction problem with symmetry constraints," *Proc. IEEE/ACM ICCAD*, pp.148-151, Nov. 1989.

[61] Alpert, C.J, Devgan, A, and Kashyap, C.V, "RC Delay Metrics for Performance Optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System*, vol. 20, no. 5, pp. 571–582, May. 2001.

[62] Ogata, M and Nishi. T, "Simple approximation models for coupled RC lines with application to delay and crosstalk estimation," *Proc. 47th Midwest Symposium on Circuits and Systems, (MWSCAS)*, vol. 1, pp. 1409-1412, 2004.

[63] I. Akrotirianakis, I. Maros, and B. Rustem. "An outer approximation based branch-and-cut algorithm for convex 0-1 MINLP problems," *Optimization Methods and Software*, 16:21–47, 2001.

[64] R. Stubbs and S. Mehrohtra. "Generating convex polynomial inequalities for mixed 0-1 programs," *Journal of Global Optimization*, 24:311–332, 2002.

[65] X. Qi, G. Wang, Z. Yu, and R. W. Dutton, "On-chip inductance modeling and RLC extraction of VLSI interconnects for circuit simulation," *in Proc. IEEE Custom Integrated Circuits Conference*, May 2000.

[66] L. T. Pillage. "Coping with RLC interconnects design headaches," *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pp. 246-253, September 1995.

[67] Atsushi Kurokawa, Takashi Sato and Hiroo Masuda, "Approximate formulae approach for efficient inductance extraction," *Proc. of the conference on Asia South Pacific Design Automation*, pp.143-148, 2003.

[68] M. Kamon, M. J. Tsuk, and J. White, "FASTHENRY: a multiple-accelerated 3D inductance extraction program," *IEEE Trans Microwave Theory Tech.*, vol. 42, pp.1750-1758, 1994.

[69] Banerjee. K and Mehrotra. A, "Analysis of on-chip inductance effects for distributed RLC interconnects," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 8, pp. 904–915, 2002.

[70] Roy. S and Dounavis. A, "Closed-Form Delay and Crosstalk Models for On-Chip Interconnects Using a Matrix Rational Approximation," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1481–1492, 2009.

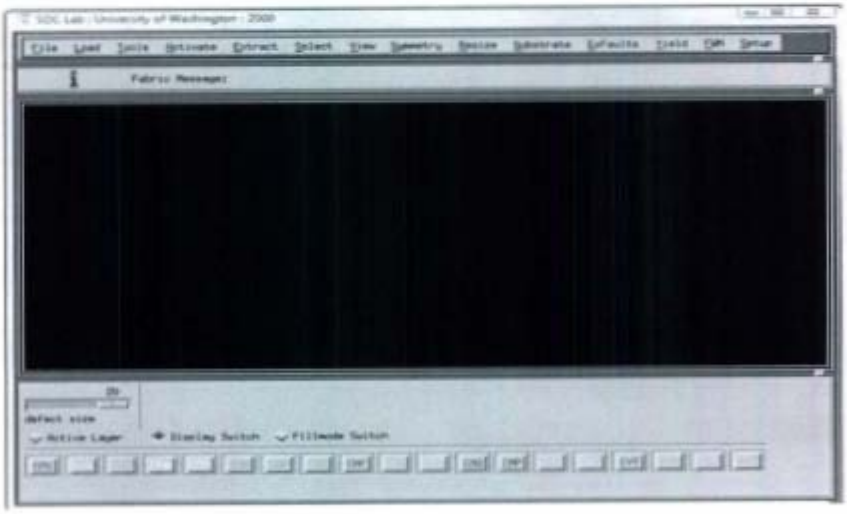
Appendix I Tutorial of IPRAIL Layout Retargeting

This appendix is a layout retargeting tutorial using IPRAIL software package implemented with the proposed MINLP retargeting algorithm. The two-stage opamp as shown in Figure 4-4 is chosen as the example layout design. This layout originally designed in a TSMC 0.25um process is migrated into a TSMC 0.18um process to meet updated performance specifications.

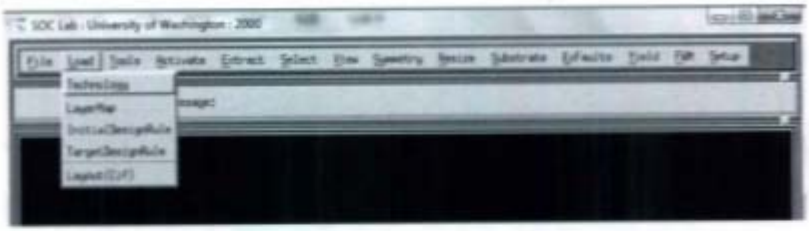
Step 1: Login *jaguar.cs.mun.ca*.

Step 2: Launch IPRAIL GUI: *>./edam &*↵

The IPRAIL GUI appears as below:



Step 3: Load the technology, layer map, initial and target design rules. The snapshot of these menu options is shown as below:



These loaded input files can be found as:

TechFile: ~/iprail/bin/demo_input/MCNC.tch

LayerMap: ~/iprail/bin/demo_input/LayerMapTSMC

InitDesignRule: ~/iprail/bin/demo_input/UWtsmc25.tech

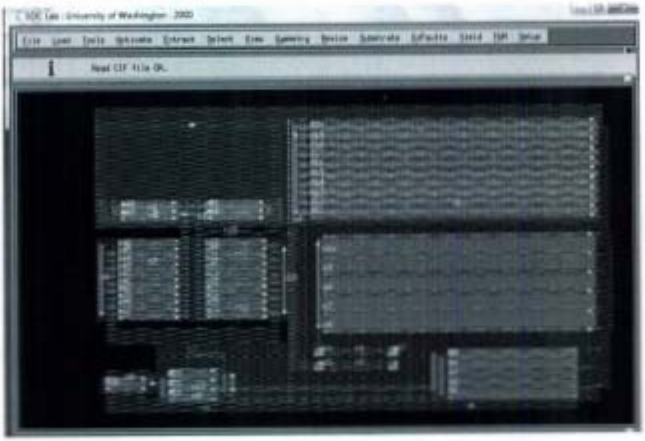
TargetDesignRule: ~/iprail/bin/demo_input/UWtsmc18_updated.tech

Note: Please make sure the files are loaded correctly by checking the command line messages.

Step 4: Load the original layout design.

Load->Layout (CIF): ~/iprail/bin/demo_input/2stage.cif.

Now, the original layout appears as shown below.



Step 5: Load piecewise performance sensitivities and maximum allowed performance deviations.

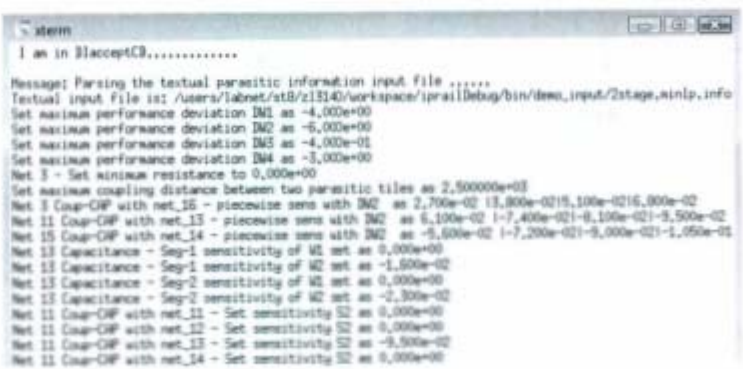
Extract->Parasitics->Read_Parasitic_Info:

~/iprail/bin/demo_input/2stage.minlp.info.

Note: This .info file is textual which specifies the performance sensitivities and maximum allowed performance deviations.

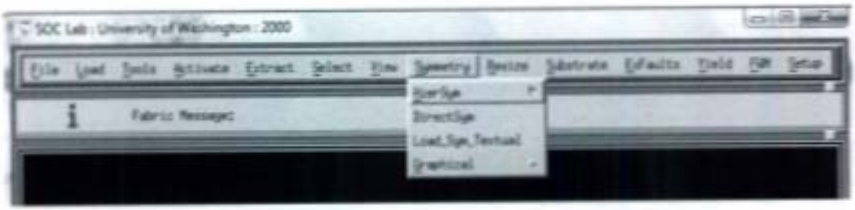
Extract->Parasitics->ExtAccToTxtFile

Here, the command line terminal displays the loaded performance deviations and performance sensitivities as:



```
stem
I am in BlacceptC9.............

Message: Parsing the textual parasitic information input file ......
Textual input file is: /users/labnet/st8/z13140/workspace/iprailDebug/bin/demo_input/2stage.minlp.info
Set maximum performance deviation DN1 as -4.000e+00
Set maximum performance deviation DN2 as -6.000e+00
Set maximum performance deviation DN3 as -4.000e-01
Set maximum performance deviation DN4 as -3.000e+00
Net 3 - Set minimum resistance to 0.000e+00
Set maximum coupling distance between two parasitic tiles as 2.500000e+03
Net 3 Coup-CAP with net_16 - piecewise sens with DN2  as 2.700e-02 (3.800e-02)5.100e-02)6.000e-02
Net 11 Coup-CAP with net_13 - piecewise sens with DN2  as 6.100e-02 (-7.400e-02)-8.100e-02)-9.500e-02
Net 15 Coup-CAP with net_14 - piecewise sens with DN2  as -5.600e-02 (-7.200e-02)-9.000e-02)-1.050e-01
Net 13 Capacitance - Seg-1 sensitivity of W1 set as 0.000e+00
Net 13 Capacitance - Seg-1 sensitivity of W2 set as -1.500e-02
Net 13 Capacitance - Seg-2 sensitivity of W1 set as 0.000e+00
Net 13 Capacitance - Seg-2 sensitivity of W2 set as -2.300e-02
Net 11 Coup-CAP with net_11 - Set sensitivity S2 as 0.000e+00
Net 11 Coup-CAP with net_12 - Set sensitivity S2 as 0.000e+00
Net 11 Coup-CAP with net_13 - Set sensitivity S2 as -9.500e-02
Net 11 Coup-CAP with net_14 - Set sensitivity S2 as 0.000e+00
```

Step 6: Load the updated symmetry requirements and new size requirements.



Symmetry->Load_Sym_Textual

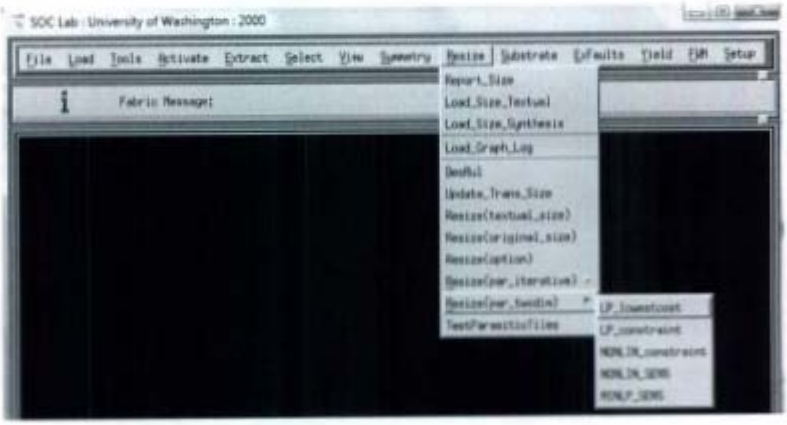
~/iprail/bin/demo_input/2stage.sym

Resize->Load_Size_Textual

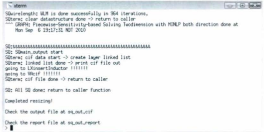
~/iprail/bin/demo_input/2stage.size

STEP 7: Conduct Retargeting.

Resize->Resize(par_twodim)->MINLP_SENS



The retargeting might take seconds up to minutes depending on the server speed and input complexity. The retargeting is finished when the command line terminal shows a success as shown below:



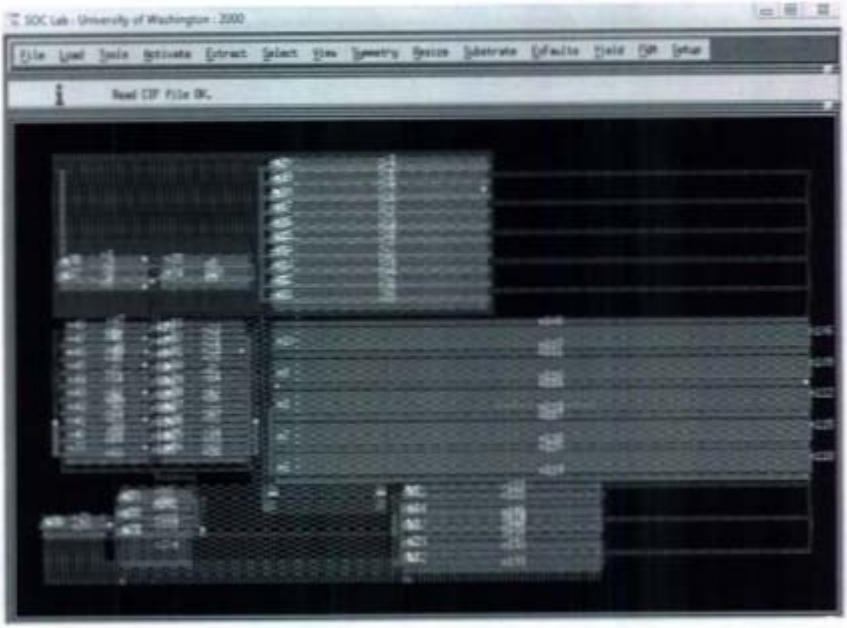
```
SQwirelength: WLM is done successfully in 964 iterations.
SQterm: clear datastructure done -> return to caller
^^^ GRAPH: Piecewise-Sensitivity-based Solving Twodimension with MINLP both direction done at
    Mon Sep  6 19:17:31 NDT 2010


SQ:&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
SQ: SQmain_output start
SQterm: cif data start -> create layer linked list
SQterm: linked list done -> print cif file out
going to LXinsertInductor !!!!!!!
going to VAcif !!!!!!!
SQterm: cif file done -> return to caller

SQ: All SQ done: return to caller function

Completed resizing!

Check the output file at sq_out.cif

Check the report file at sq_out.report
> ▌
```

Step 7: Review properties of target layouts, retargeting time statistics as well as IPOPT solving information. The information includes extracted parasitics from target layouts, IPOPT solutions of target layout geometries. IPOPT input scripts and IPOPT solving process log.

The target CIF layout is generated in a CIF file:

~/iprail/bin/resize_report/sq_out.cif.

Loading this file into IPRAIL, the target 2stage layout appears as below:



You can check the time statistics in:~/iprail/bin/resize_report/timestamp.txt as:

```
xterm
> more timeStamp.txt
Resize start time: Mon Sep  6 19:16:43 MDT 2010
Scan-line end time: Mon Sep  6 19:16:53 MDT 2010
End of scan-line: Mon Sep  6 19:16:54 MDT 2010
Start of IPOPT: Mon Sep  6 19:17:03 MDT 2010
End of IPOPT: Mon Sep  6 19:17:26 MDT 2010
End of core problem: Mon Sep  6 19:17:31 MDT 2010
Resize end time: Mon Sep  6 19:17:32 MDT 2010

>
```

Check the extracted parasitic resistance/capacitance/inductance in:

~/iprail/bin/resize_report/parExtrRpt.txt as:

```
xterm
> more parExtrRpt.txt

*** Report of extracted parasitics ***

-----------------------------------
Net 3    has    2 tiles.
Tile  653 CMF H
      (   14150,    21440) (   18960,    21760)
Parasitic-Tile Resistance: 1.677000, Parasitic-Tile Capacitance: 5.500560e-16

  Parasitic-Tile Inductance: 5.87751e-05
Tile  633 CMF V
      (   13600,    19920) (   14150,    21760)
Parasitic-Tile Resistance: 1.066000, Parasitic-Tile Capacitance: 2.485420e-16

  Parasitic-Tile Inductance: 2.50875e-05
Netsplit  631 CMF V
      (   13600,    19520) (   14150,    19520)
Netsplit Resistance: 0.039000, Netsplit Capacitance: 9.093000e-18
MFT Resistance: 0.000000, MFT Capacitance: 0.000000e+00
Parasitic-Net Resistance: 2.782000 ohms, Parasitic-Net Capacitance: 8.076910e-16 F

Parasitic-Net Inductance in nH: 0.000084

-----------------------------------
Net 4    has    2 tiles.
Tile  695 CMF V
      (   21200,    21200) (   21760,    29080)
```

These extracted net parasitics from the target layout can be further used to perform post-layout simulations. Moreover, you can also check the solver input script, the solution file, and the solving process log by access the following textual files:

~/iprail/bin/COIN_optimization/ TD_par.SIF

~/iprail/bin/COIN_optimization/TD_par.SIF.rpt

~/iprail/bin/COIN_optimization/ COIN.log

Appendix II List of Author's Publications

[1] Zheng Liu and Lihong Zhang, "Performance-constrained template-driven retargeting for analog and RF layouts," *in Proc. IEEE/ACM Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 429-434, 2010.

[2] Zheng Liu and Lihong Zhang, "A performance-constrained template-based layout retargeting algorithm for analog integrated circuits," *in Proc. Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 293 - 298, 2010.

[3] Zheng Liu and Lihong Zhang, "Performance-constrained parasitic-aware retargeting and optimization of analog layouts," *Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, St. John's, Newfoundland, May 2009.

[4] Zheng Liu and Lihong Zhang, "Optimization of Parasitic Constraints for Analog Integrated Circuits," *Proceedings of Newfoundland Electrical and Computer Engineering Conference (NECEC)*, St. John's, Newfoundland, Nov. 2008.

[5] Lihong Zhang and Zheng Liu, "Directly performance-constrained template-based layout retargeting and optimization for analog integrated circuits," *Integration – The VLSI Journal*, pp. 18-31, Oct. 2010.