Layout Effects Modeling for Analog Circuits Design Automation

Henry Hoi Yun Chan

Doctor of Philosophy

Department of Electrical and Computer Engineering

McGill University

Montreal, Quebec

2008-8-31

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy

Your file   Votre référence
ISBN: 978-0-494-50794-0
Our file   Notre référence
ISBN: 978-0-494-50794-0

DEDICATION

To my dearest Mother, Father, Winnie and Angie

ACKNOWLEDGEMENTS

I am grateful to have Prof. Zeljko Zilic as my research supervisor, who introduced me into microelectronics research at my undergraduate years and provided the equipment and stimulating research environment in the IML at McGill. For the past years, he followed my research closely and critically. His visionary thoughts constantly provided me with inspirations, research techniques and the proper perspective at numerous stages of this work. His encouragements are especially valuable at certain critical points in this research.

I would like to thank Prof. Roni Khazaka and Prof. Dennis Giannacopoulos for their valuable feedback and to be my thesis committee members. Although my name is the only one that is printed on the thesis cover, several individuals contributed either directly or indirectly in bringing this work to its conclusion. I would like to thank Atanu Chattopadhyay, Jean-Francois Boland, Bojan Mihajlovic, Marc Boule, John Storey Bishoff, Robert Armstrong, Terry Thomas and Trent McConaghy for their inspiring discussions, critiques and comments. I thank Stephan Bourduas, Jean-Samuel Chenard, Kahn-Li Lim and Nathaniel Azuelos for the collaborations, excellent French and software support, and Jason McLaren, Zuweina Sood for tools support. I also thank all my past and present colleagues, staff and students in MACS and IML of McGill University, Analog Design Automation Inc., and Synopsys Inc.

Finally, no word would be enough to show my appreciation towards my girlfriend, Angie, for her tremendous sacrifice all along, to be always there when I needed the most. Her unwavering support and patience are the energy sources that propelled me since the beginning to the end of this long, dis-orienting and often difficult journey. Last but not least, every trait that I have, I owe it to my dearest parents and my sister, Winnie, who taught me, encouraged me and supported me since my early years to pursue my dreams, and not to give up without trying my best.

ABSTRACT

Deep sub-micron (DSM) integration brings about aggressive technology scaling to accommodate large and high-speed systems onto a single chip. The integration and migration of mixed-signal systems to smaller process nodes have shortened the traditional analog circuit design cycle and increases performance influence due to parasitic coupling. Current analog optimization tools demonstrate promising results at the schematic netlist-level, but inherited layout effects are excluded until the physical design is completed. If coupling effects are ignored or poorly modeled, schematic optimization results are no longer accurate with respect to silicon measurements. On the other hand, physical design tools are traditionally guided by geometric constraints. In interconnect-dominated designs, both parasitic-aware circuit optimization and performance-driven physical design are crucial for rapid design closure. This thesis addresses both issues by integrating schematic optimization and the physical design process. Through the use of virtual interconnect parasitic models and light-weight parasitic models, the simulation-based circuit optimizers and placement tools can exchange design and performance information while operating at their full capacities. The interconnect models also provide provisional routing configurations. A novel compaction process for analog layout further refines the block and interconnect positions with respect to DSM effects.

ABRÉGÉ

Les technologies d'intégration submicrométrique profonde (DSM) permettent la réalisation de systèmes complexes à haute vitesse et sur une seule puce. La migration de systèmes à signaux mixtes vers des procédés de fabrication à plus petite échelle raccourcit le cycle de conception traditionnel des circuits analogiques et augmente l'influence des effets parasitiques sur la performance des circuits. Les outils d'optimisation de liste des interconnexions présentent des résultats prometteurs, mais les effets topologiques intrinsèques sont exclus jusqu'à la réalisation physique. Si les effets de couplage sont ignorés ou s'ils sont mal modélisés, les résultats obtenus lors de l'optimisation de la réalisation topologique ne refléteront pas précisément la réalisation physique. Par ailleurs, les outils automatisant la génération de la réalisation physique et du masque de fabrication du circuit sont traditionnellement guidés par des contraintes géométriques. Pour les réalisations dominées par les interconnexions, une optimisation basée sur les effets parasitiques et sur la performance de la réalisation physique est de très grande importance, pour faciliter la conclusion rapide des projets. Cette thèse adresse les deux points précédents en intégrant l'optimisation des diagrammes schématique avec les procédés de génération du dessin physique. En utilisant des modèles virtuels et simplifiés des effets parasitiques, les logiciels d'optimisation basés sur la simulation peuvent donc communiquer avec les outils de placement sans affecter la performance de la simulation. Les modèles d'interconnexion fournissent aussi des configurations provisionnelles sur le tracé. Une nouvelle approche au processus de compaction, applicable aux topologies analogiques, permet un meilleur raffinement du positionnement des interconnexions en tenant compte des effets DSM.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AC ...	Alternate Current
BEM ...	Boundary Element method
BJT ...	Bipolar Junction Transistor
CAD ...	Computer-Aided Design
CDR ...	Clock Data Recovery
CI ...	CMOS Inverter
CMOS ...	Complementary Metal-Oxide Semiconductor
CP ...	Charge Pump (PLL component)
DA ...	Differential Amplidiers
DC ...	Direct Current
DSM ...	Deep Sub-Micron
ECO ...	Engineering Change Order
EDA ...	Electronic Design and Automation
EM ...	Electro-magnetic
FEM ...	Finite Element Method
HDL ...	Hardware Description Language
IC ...	Integrated Circuit

KL ... Karhunen-Loève

LF ... Loop Filter (PLL component)

MEMS ... Micro-Electro-Mechanical Systems

MHS ... Maximum Horizontal Strip

MOSFET ... Metal-Oxide Semiconductor Field Effect Transistor

MVS ... Maximum Vertical Strip

OpAmp ... Operational Amplifier

OPC ... Optical Proxmity Correction

OPT ... schematic optimization (Co-optimization flow)

PD ... Phase Detector (PLL component)

PDF ... Probability Density Function

PDK ... Process Design Kit

PLC ... Placement (Co-optimization flow)

PLL ... Phase-Locked Loop

PMD ... Parasitic Macro-Device

PSS ... Periodic Steady State

RF ... Radio Frequency

RMS ... Root Mean Square

RTE ... Routing (Co-optimization flow)

RTL ... Register Transfer Level

SA ... Simulated Annealing

SIM ... Circuit Simulator (Co-optimization flow)

SoC ... System-on-Chip

SSN ... Simultaneous Switching Noise

UI ... Unit Interval

VCO ... Voltage-Controlled Oscillator

VI ... Virtual Interconnect

VLSI ... Very Large Scale Integrated Circuit

CHAPTER 1

INTRODUCTION

The exponential increase in the density of transistors on a silicon chip with time has been a prominent trend in the semiconductor industry. Gordon E. Moore predicted in 1965 that transistor count would increase by a factor of 2 for every 18 months. This is confirmed repeatedly by numerous surveys conducted by *International Technology Roadmap for Semiconductors* (ITRS) [10] and is now known as the *Moore's Law*. It serves as the well-respected basis for predictions of future technology growth trends. The growing number of smaller devices implies that if the accompanying issues of increased design complexity and defects affecting production yield can be overcome without significant cost increase, continual functional and performance improvements of a chip are possible at the same cost. Examples are new system-level design paradigms to take advantage of improved performance, better design and testing tools to handle increasingly complex and dense circuits, lithography techniques with higher resolutions and geometric correction techniques to compensate for optical diffractions, etc.

To reduce packaging cost and long-term reliability issues, digital and analog circuits are often integrated together onto the same silicon die. While such *mixed-signal* systems provide improved performance as packaging and some interconnect parasitics are minimized, new interference issues arise. Analog design must adapt and maintain its robustness towards this new operation environment in order to preserve the data quality of the overall system.

Electronic design automation (EDA) tools are software applications that aid designers in the productivity of designing complex *integrated circuit* (IC). Until recently, the majority of EDA research and development activities were targeted towards digital designs. Even though more and more circuit functionality is being performed in the digital domain due to its superior robustness against noise and interference, certain analog circuits, such as filters, amplifiers, analog-digital converters, PLL, etc., are irreplaceable and remain an integral part of most systems. On the contrary, advancements towards *ubiquitous computing* [71] in the fields of wireless and *micro-electro-mechanical systems* (MEMS) sensor technology will continue to propel the need of analog integration into the foreseeable future. With reduced product lifecycle, shrinking fabrication minimum feature sizes and increasing clock rates, a successful EDA flow must achieve rapid design closure amid strong parasitic influences. Hence, an effective analog EDA flow is crucial for any modern mixed-signal designs to meet the required overall performance objectives efficiently.

A typical IC design cycle involves the following steps:

- System design and verification
  - Functional design, partition and verification
- Circuit design and verification

  - circuit topology selection
  - component selection and simulation
  - Circuit verification and noise analysis
- Physical design and verification
  - Floorplanning, die size estimation and packaging selection
  - Block placement
  - Global and detailed routing
  - Parasitic extraction and simulation
  - Post-extraction verification and noise analysis
- IC fabrication, test, packaging
- Product testing

Each of the above steps can be done manually, automatically or semi-automatically with designers and design aids. The order of these steps are based on a *top-down* hierarchy of models that become progressively more complex. In summary, preliminary functional correctness is first addressed using simple models to simplify the design problem. The design process is repeated again at the next lower level, as more detailed elements and thus design freedom are introduced. Although design processes can occur at functional, circuit and physical levels in the general flow, most of the design processes are performed primarily at one level. The abstraction level where the majority of design processes take place is dictated by the size and complexity of the target system. In the modeling point of view, the predominant IC design paradigm takes the following pattern:

- **Design**: Perform design processes at the modeling level with the simplest circuit models (*highest* modeling level) that is practical and effective to cover the system properties concerned.
- **Implementation**: Design refinement towards the physical level with refined circuit models using newly obtained information. Verify if performance is within required specifications.
- **Verification**: Compare refined performance measures with required specifications. Additional performance *margins* may be needed to accommodate subsequent performance degradations due to unencountered effects.
- **Re-iteration**: If performance degradations exceed margins, back-track to the previous level and attempt an alternative design. The process ends when the performance of an implementable design satisfies the required specifications.

Since lower-level effects are absent from current-level models, the decision of designing circuits at a certain abstraction level implicitly determines which types of effect are significant enough to be revealed and examined and which ones are left to be absorbed by performance margins and verified afterwards. This imposes an important efficiency-accuracy trade-off.

For example, digital systems with billions of gates are designed at the system level. Through abstractions using *Boolean algebra* and various *hardware description languages* (HDLs), large systems can be designed efficiently from the top system level to the *register-transfer level* (RTL) down to the gate level. This system is then *verified* at the circuit and physical levels, via synthesis tools. Performance degradations due to lower-level effects can usually be absorbed by design margins. This is possible due to the wide noise margin.

Otherwise, local design refinements that remedy the violations usually suffice. This is possible due to well-defined signal boundaries between subsystem interfaces. On the other end of the spectrum, *radio frequency* (RF) circuit design requires very detailed modeling of each component used. However, due to its small size, they are often designed directly at the schematic and physical levels.

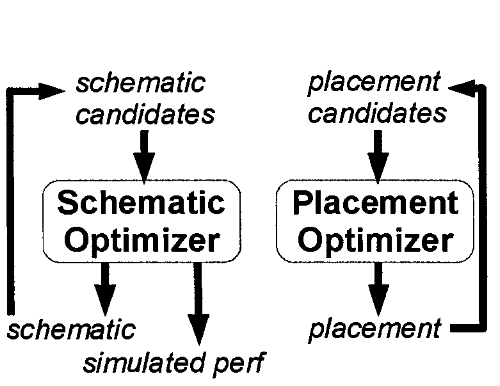
Curiously, as much as they are different, digital and RF designs share a significant property in the interest of layout parasitic modeling, the *regularity* of the design topology. Dimensions and spacings of digital layout are highly regular in order to facilitate efficient automatic generation. Its noise robustness affords the use of suboptimal, simplified layout design. RF designs also have high degree of conformity from one version to another, due partly to the small number of components and the reliance of test data from previous versions.

Parasitic modeling benefits from predictability of the layout parasitics of both design types. For example, wire-length delay correlations, analytical computations of bus wires crosstalk and reflection [61] are possible due to the uniform widths and signal characteristics. *A priori space mapping* [72] of layout-based RF design is practical due to the layout and parasitic properties being known.

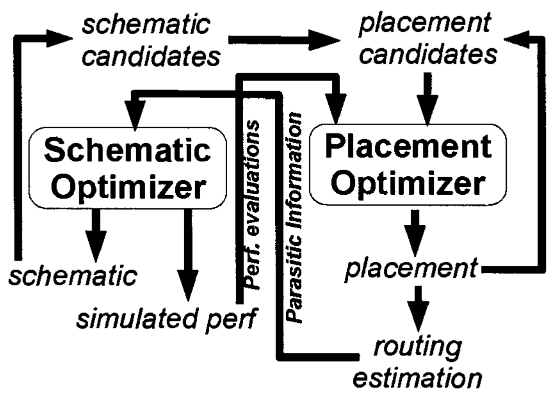
Analog circuit design is usually performed at the circuit, or *schematic*, level with simplified circuit models, which do not adequately capture its layout effects. High performance analog circuits require optimized physical design to attain their performance requirements. It is also too complex for equation-based parasitic modeling except for the very smallest designs. As the differences in actual and predicted performance remain unknown, the limited error margins are maximized to increase design closure probability, and as a result, counteracting the performance advantage offered by the new technologies. Hence, predicting the layout effects of an irregular and noise sensitive analog circuit design remains a formidable challenge. A better approach is needed in the age of aggressive design specifications where performance margins come at a premium.

1.1 Problem Description

Simulation-based device sizing optimization has been proved successful in efficiently exploring the circuit design space at the schematic-level and delivering a near-optimal solution. As the IC industry continues to adopt advanced fabrication technologies with progressively smaller minimum feature sizes, circuit performance becomes strongly influenced by layout-dependent parasitic effects, in additional to device sizings. This compromises the effectiveness of schematic-level optimization since a well-performing schematic design does not imply robustness towards layout effects. Unfortunately, formal extraction of these parasitic effects is not possible until the physical design is finalized. A new framework that efficiently furnishes layout effects of the candidates during the circuit design process is badly needed. Rutenbar [55] stated that the key criteria for the next-generation analog design automation tool are: (i) the integration of independent point tools, (ii) design constraints extraction, management and reuse, and (iii) system-level design optimization and exploration. This thesis contribute novel concepts and provisions towards solving these issues.



(a) Independent schematic and placement optimization flows



(b) Integrated flow

Figure 1–1: Schematic and placement optimization

1.2 Original Contributions

This thesis is based on a series of published research papers on incorporation of parasitic effects in analog design automation [22], [12], [23], [24], [21], [20], [19]. To enable layout-aware circuit optimization for *full-custom* analog designs, this research proposes the integration of currently independent circuit and layout optimization steps (Fig. 1–1(a)) to enhance the convergence among the optimized circuits and their post-extracted performance in the presence of layout parasitic effects. The challenges for next-generation analog design automation tools [55] in the DSM era are addressed in this thesis through careful extraction, exchange and reuse of parasitic and performance information between the circuit and layout optimization loops. As illustrated in Fig. 1–1(b), parasitic information from layout optimization is feedback to the schematic optimizer, in exchange for the performance information of the layout. The improved quality and efficiency of the integrated loop make system-level accurate performance space exploration possible. None of the currently offered optimization techniques encompasses layout effects without involving proprietory cell libraries, thereby compromising design flexibility.

Through a tool-independent and library-independent circuit and physical design co-optimization methodology, the optimization flow proposed here incorporates parasitic estimations in the optimization loop by extrapolating extracted parasitic values according to anticipated layout modifications. In return, the physical design flow reuses the simulation results for performance-driven layout optimization. Fig. 1–2 illustrates the implementation of the proposed parasitic-aware design automation methodology, compatible with off-the-shelf EDA components. Through information exchanged by sharing database and intermediate results, both schematic and layout optimization routines mutually benefit from each other's optimization results and thus improve their effectiveness. On the left side of Fig. 1–2, the simulation-based optimization flow tunes device sizes on the schematic at the presence of layout-effect parasitics. Layout effects are taken into account through annotation of parasitic effects derived from layout optimization. Optimized circuit parameters and performance data are then fed into the physical optimization flow on the right side.

1.3 Claims of Originality

The thesis has contributed the following new concepts and techniques:

Parasitic-Aware Analog Schematic Optimization Algorithm [21][23]

Existing simulation-based schematic optimization engine tunes performance based solely on device sizing. This thesis proposes the back-annotation and estimation of layout parasitics through computation of the block displacement and the change of interconnect geometry of a preliminary layout due to device sizing changes. Layout changes are modeled after the behavior of human designers to yield realistic results. The method has been

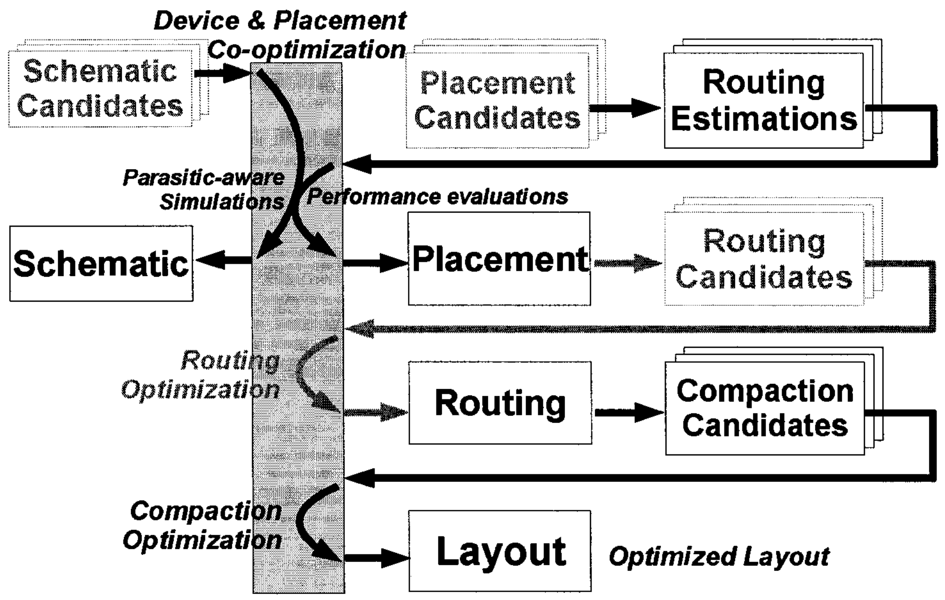


Figure 1–2: Optimization-based circuit and physical design flow integration

adopted by research groups for *Circuit Explorer* of *Synopsys Inc.* and *Creative Genius* of *Analog Design Automation Inc.*.

Parasitic-Aware Analog Layout Compaction Algorithm [24]

This thesis proposes a performance-driven, parasitic-aware compaction algorithm for analog layout. Current methods are mostly geometric-driven without DSM parasitic insights, such as substrate coupling. The proposed compaction optimization is carried out to fine-tune block placement while balancing the often opposing substrate coupling and interconnect parasitic effects.

Parasitic Macro-Device Concept [21][24]

This thesis proposes a novel concept that treats wires and interconnect in a circuit as "devices". An algorithm identifies clusters of parasitic devices in the post-extraction netlist and encapsulate each of them as an n-port Parasitic Macro-Device. Identity of the corresponding wire or interconnect is stored in its property list. Consequently, the topology of the modified post-extraction netlist circuit graph is consistent from one extraction to another, which enhances the ease of model-building process for parasitic estimation.

Simultaneous Switching Noise Models [22]

Simulation of DSM performance of high-performance mixed-signal systems such as *system-on-chips* (SoCs) often involves transient excitations of large number of circuits in order to generate the realistic switching noise level during nominal operations. In Chapter 5, a reduced-order chip-level simultaneous switching noise model is proposed. Model order reduction is achieved through the novel application of *Karhunen-Loève* transformation [38] to capture and reproduce the essence of noise signatures from available noise samples.

Cyclostationary Jitter Model [19][22]

PLL jitter simulation is time-consuming, due to the small time resolution needed relative to the simulation time interval. A compact model is derived to estimate the periodic steady-state PLL jitter due to substrate coupling and external interference by exploiting the temporal correlations between noise and the clock signal. This work is cited by [60].

1.4 Thesis Outline

In Chapter 2, fundamental information about parasitic effects in silicon *integrated circuits* (ICs) is given. Resistive, capacitive and inductive parasitic effects are introduced. Several practical approaches, based on either the *boundary element method* (BEM) or the *finite element method* (FEM) are explained to extract these effects in ICs. In Chapter 3, the basis of various design automation schemes are presented. Manual schematic-level design are replaced by optimization-based circuit design methodologies. Complex floorplanning and routing problems also served by search-based physical design automation tools. Parasitic effects, however, are not dealt with until after the physical design process is completed. In Chapter 4, a co-optimization strategy that links schematic and layout optimizations together is presented. The layout optimizer provides accurate parasitic information to the schematic optimizer, while the schematic optimizer gives quantitative performance evaluation results to aid the physical design process. This exchange of crucial information is a major contribution to the improvement of design closure efficiency under the influence of parasitic effects in an automated flow. A detailed example using this proposed co-optimization methodology is presented in Chapter 5. In this chapter, circuit and physical design options are optimized via PLL jitter analysis over design parameters, SoC module and noise barrier placements using *periodic steady state* (PSS) analysis. The scope of design considerations including the effects of aggressive system integration and guardband configurations exceeds the capabilities offered by conventional computer-aided design tools. Finally, the conclusions and implications of this work are discussed in Chapter 6.

CHAPTER 2

FUNDAMENTAL CONCEPTS AND GROUNDWORK

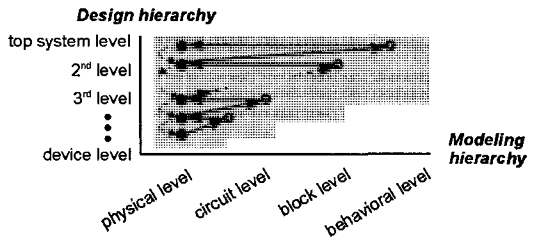
This chapter consists of two parts. The basic concepts of the conventional circuit design flow, regarding layout-related parasitic effects, the extraction process and how they impact the effectiveness of the IC design flow will be represented. Next, the contributions and implementation of essential components for the proposed co-optimization flow, such as parasitic modeling for placement and routing, sensitivity analysis, representation methods of circuit and the physical design will also be discussed thoroughly.

2.1 Analog Circuit Design

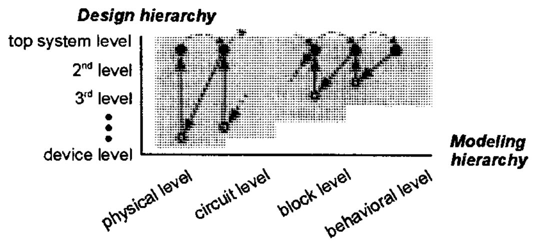
Analog circuitry plays an important role in interfacing between real world information and with signals of complex digital systems. Deep sub-micron (DSM) integration brings about aggressive technology scaling to accommodate sensitive analog and complex digital systems onto a single chip. The mixed-signal systems gain performance advantages by aggressively adopting the latest digital processes. New issues affecting analog circuit development include rapid fabrication process migration, and the sharing of a typically noisy operating environment with digital systems. Specifically, the traditional analog circuit design cycle is being shortened, while the interconnect, packaging and substrate parasitics greater contributions to performance degradation.

A circuit *topology* is the interconnection of subcircuits and elementary components such as *Bipolar Junction Transistors* (BJTs), *Metal-Oxide Semiconductor Field-Effect Transistors* (MOSFETs), *resistors* (R), *capacitors* (C) and *inductors* (L). Traditional analog circuit design begins by drafting the schematic-level topology of the system and then determining the design variables for all components. Circuit design at the schematic level provides good balance between prototyping ease and simulation accuracy. Depending on the size of the system, it can be partitioned into several subsystems, where each part is determined independently. Complex systems can be built hierarchically by encapsulating subcircuit blocks with well-defined interface ports. The topology defines a schematic model of the circuit that designers rely on to evaluate characteristics of the circuit and assign device sizes. The schematic design process is usually carried out through a front-end graphical user interface, where the simulation tool extracts the underlying circuit *netlist* for simulation. The *device sizing* of each subsystem at each hierarchical level subcircuit is then performed. The result is a *sized* schematic of the system.

To simplify a complex circuit design problem, the system concerned is divided into a number of parts, or *subcircuits*. The *design hierarchy* is a set of system partitioning



(a) A bottom-up design flow



(b) A top-down design flow

Figure 2–1: Design and modeling hierarchy in IC design flows

schemes that ranges from a collection of subcircuits containing a single device at the bottom level, to containing the entire system at the top. To preserve computation efficiency, modeling details are moderated according to the subcircuit scope. The amount of electromagnetic phenomena involved with even a small subcircuit can be daunting [48], hence only the most significant phenomena are considered if the model scope is large, whereas a more detailed circuit model can be used if the subcircuit under consideration is small. The array of models that describe circuit behaviors from the coarse behavioral to accurate physical models is known as the *modeling hierarchy*. Fig. 2–1 shows how the design and modeling hierarchies interact in the IC design flow. While the modeling hierarchy is always traversed in a top-down manner (right to left in Fig. 2–1), the design hierarchy can either be traversed from the bottom device level up to the top system level, or vice versa.

The *bottom-up* approach starts at the bottom-left corner in 2–1(a) with elementary components or small component groups created independently and then assembled together at a higher design hierarchical level. It progressively selects from the available design options and if these subcircuits pass the evaluation criteria, they are propagated to the next level. At the top level, the completed system is verified. The advantage of this approach is that from device model and sizing parameters upward, lower level details are known and used in the evaluation at each level, hence no performance approximations or assumptions are needed. Its major drawback is that the overall performance of the eventual system cannot be easily verified and tracked during the intermediate stages with a set of unconnected subsystems.

The *top-down* approach in 2–1(b) begins with the top-level system behavioral model at the upper-right corner. The model is then replaced by a number of lower-level block that satisfy the respective performance specifications. The interface and specifications for each lower-level blocks are then assigned. This refinement process is repeated until all lower-level blocks are replaced by elementary components. While an intact system model is always present during each stage of the model refinement process, these models necessitate performance assumptions in the modeling of lower-level details before they are known. Piecewise functions can model certain non-linear effects within some limits. Exceeding them, accurate models for highly non-linear effects become expensive while evaluations based on simple models can be misleading.

The differences between the top-down and bottom-up flows explain why the top-down design style is effective in digital systems, and that it causes considerable difficulties when applying directly to analog circuits, due to the existence of complex dependencies among design variables and performance measures. Although the bottom-up approach is effective when subcircuit behaviors are inter-related with unknown or ill-defined interfaces, analog and digital component must be refined and verified in tandem in mixed-signal system design. Estimated parasitic-aware circuit models are thus needed to enable a top-down analog flow.

2.2 Parasitic Effects in Integrated Circuit Design

For silicon fabrication, more details of the physical design of the circuit, such as the specific location, shape, dimension and material for every component and interconnect wire, are needed. It begins with floor-planning, followed by determining the dimensions of subsystem blocks. This is then followed by arrangement of the packing configuration of the blocks and the placement of device layout on the die. Finally all necessary nodes are connected corresponding to the given schematic. These steps must all be carried out respecting the given physical design rules and some matching and symmetry constraints, with considerations regarding the consequential fabrication random process variations. Parasitic coupling inherited from the design are extracted from the physical design and evaluated against the required specifications, to ensure that it has not been degraded from that of its schematics. The evaluated information can be used to refine the physical design. Compared to schematic design, physical design is time-consuming, and will typically be performed after its schematic-level performance is satisfactory.

Electrical circuits are interconnections of *electronic devices* that implement certain functions through the interactions of voltage and current among the electrical nodes. Circuits are usually described textually by the connection *netlist* or a graphically by *schematic* drawing. With conventional *very large scale integrated circuit* (VLSI) fabrication technologies, the interconnects are routed through a stack of connected planar metal layers. Below, transistor devices are embedded in, or are on the surface of the silicon substrate, which provides mechanical support to the circuit. The geometric design of this physical implementation is commonly known as the *layout* of the circuit. Parasitic effects are the consequences of the particular implementation of the electronic circuits. This includes the specific geometry and material choice of the respective layout. These layout-dependent parasitics are created by a special program, the *parasitic extractor*, through examination of the geometry and interaction among all layout shapes. Exact details of the extraction are governed by the *extraction rule deck*, which is a set of rules and other specifications, found in the *process design kits* (PDK). The rule deck varies for different fabrication processes, and changes within a process when either the process properties change or certain parasitic models are refined. The extracted parasitic devices are inserted into the schematic netlist, forming the extracted parasitic netlist or simply the *extracted netlist*.

In the context of this thesis, the term *parasitic* applies to interactions among circuit nodes that do not belong to the intended *idealized* schematic design. It includes devices inserted by designers representing the estimated parasitic effects. On the other hand, designers often rely on parasitic effects to implement passive devices. They are considered intentional devices instead of parasitics. Whenever there are potential differences between any two conductors in a network, an electric field is present between them. Analogously, a magnetic field is present around any conductor that carries current. The strength of the fields attenuate over separation distances at rates that corresponding to the properties of the insulating material. Ideally, interconnects are perfect conductors, while disconnected devices and nodes in the circuit are completely insulated from each others. In VLSI, insulations are provided by poor conductors, such as silicon dioxide, air gaps, as well as by means of reverse-biased *p-n* junction potentials. They reduce most but do not completely eliminate induction among signals. Besides, interconnect wires have small but non-zero resistances that hamper signal propagations. These physical properties are generally known as *layout parasitic effects*. Their consequences of the implemented circuits are departures from predicted behaviors of their schematic models. However, these parasitic effects also provide several means to integrate on-chip passive devices, such as resistors, capacitors and inductors.

At frequencies greater than a few GHz, the amount of self-inductive effects and signal reflections due to impedance mismatch becomes significant [6]. In these cases, the parasitic effects can be modeled by inline impedances containing inductors. For most analog applications, the operating frequency is usually lower than a few GHz and parasitic inductive effects can be safely ignored.

During circuit operation, a small but measurable amount of charge can leak from one node to another through the insulation. This diversion of charge following the signal paths is known as *leakage current*, and is modeled by auxiliary conducting paths using parasitic resistors. Although the magnitude of leakage current is negligible at its sources in most cases, the combined effect looms as a critical factor with a large number of leakage sites. For instance, consider a multi-million gate microprocessor chip fabricated in 130nm technology with supply voltage of 1.2-1.3V, the total leakage current constitutes 10-30% of its active power [54]. At the 70nm node with supply voltage less than 1.0V, over 50% of the overall power dissipation is due to leakage current [54].

For long wires, signal levels may decrease with distance travelled due to resistance of wire material as it travels further from the driving gate. This effect is known as *IR drop*, and can be modeled by an inline resistor. Along neighboring wires, signal activities may also couple voltages or induce currents across neighboring conductors. These are commonly known as *parasitic coupling*. EM fields are ubiquitous, hence a multiple-port network is needed to describe the parasitic effects between every node pair in general. In

Table 2–1: Physical Orientations of Parasitic Devices

Parasitic Type	Orientation
Interconnect resistance	inline
cross coupling capacitance	lateral
substrate coupling near surface	lateral
Interconnect capacitance to ground	vertical
well capacitance	vertical
substrate bulk to bulk node resistance	vertical

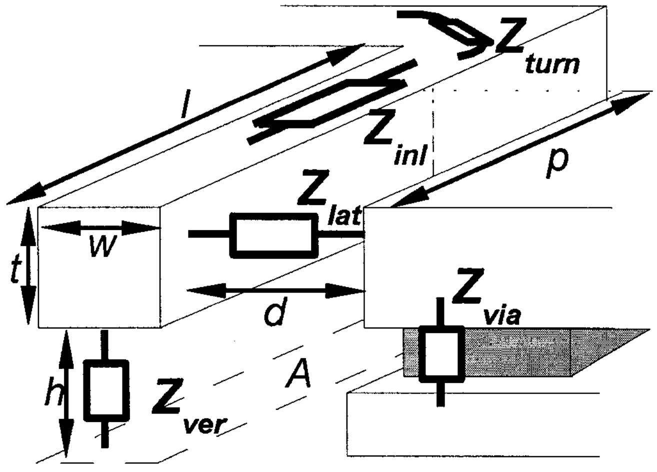


Figure 2–2: Interconnect parasitic models for optimization

practice, the parasitic extractor would determine the significant parasitic effects of the layout, and annotate the circuit netlist with 2-port passive devices. Fig. 2–2 illustrates generic examples of these parasitic device types. They can be modeled by parasitic capacitors and inductors. Layout parasitic effects can be classified into *inline parasitics* that take place *along* the direction of intended signal propagation, *lateral parasitics* that take place *across* the intended signal paths, and *vertical parasitics* that are *perpendicular* to the substrate surface. They respond to layout changes differently. Table 2–1 lists the classification of common parasitics found in CMOS.

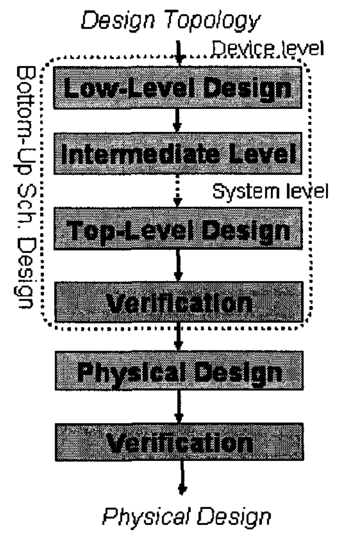
Fig. 2–2 illustrates the generic view of several parasitic device types. $Z_{inl}(l, w)$ includes interconnect resistances and self-inductances along interconnect segments, it varies according to the length l and width w of the interconnect segment. $Z_{lat}(p, d)$ refers to *lateral* cross-coupling capacitances and mutual inductances among neighboring objects of the same layer, it varies with their separation distances d and overlapped length p. $Z_{ver}(A)$ denotes *vertically*-oriented capacitance, inductances between interconnects of different layers. Ignoring the fringing effects [61], the capacitance is proportional to the projected area A. Layer thickness t is fixed throughout layout synthesis. Finally, $Z_{turns}(w)$ and Z_{via} model respectively the additional impedances for turns and via in the interconnect. These parameters are usually available with the fabrication PDKs. Otherwise, a layout test set containing straight and one-turn wires of variable widths can be set up once for each process technology to obtain the necessary parasitic model parameters Z_{inl}, Z_{lat}, Z_{ver}, Z_{turns} and Z_{via}. Additionally, parameter sensitivities with respect to geometric dimensions $\frac{\partial Z_{inl}}{\partial l}$, $\frac{\partial Z_{lat}}{\partial p}$ and $\frac{\partial Z_{ver}}{\partial A}$ are also derived for parasitic estimation purposes. Extracted values and sensitivities of resistances and capacitances of various interconnect widths and lengths are discussed in Section 2.5. Discussion of substrate and interconnect discretization for parasitic modeling follows next.

The parasitic effects shown above are functions of physical design geometry and fabrication material characteristics. They may also be functions of signal frequencies. If so, the interconnection is also known as a *dispersive* channel in the field of communication systems. In the era of DSM technologies, spaces separating wires and devices diminish, signal frequency increases and the amount of parasitic coupling increases. Accurate extraction of parasitic parameters is thus an important step in circuit design. The following sections explain the details of parasitic extraction from physical design geometry.

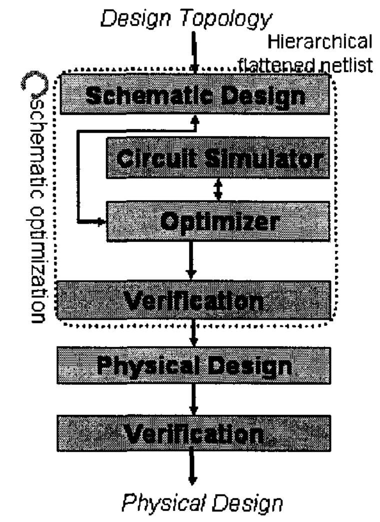
2.3 Parasitic Effects in Physical Design

One of the many objectives that constitutes a quality fabrication process is to minimize the performance degradation for all circuits due to their inherited layout parasitics. Sources of performance degradation can be attributed to general layout parasitic effects and fabrication *process variations*. Parasitic effects have been discussed earlier in this chapter. Process variation is due to the systematic and stochastic irregularities on different parts of the wafer causing variations of physical properties. Stochastic irregularities include bleeding, diffraction, thickness and thermal gradients [32], patterns and proximity effects [25]. They are beyond the control of designers, which must rely on accurate models and quality control of the foundry. Systematic process variation can be minimized through robust layout design practices, such as common centroid, inter-digitation and symmetry, etc. [32]. For example, device mismatch is the process that causes time-independent variations of geometric discrepancies of otherwise identically designed devices [50]. Fig. 2–3(a) depicts such design flow. The schematic design begins at the device level, the layout design and parasitic extraction then follow.

Parasitic devices in a post-extracted netlist can be separated into two types, (i) those that are present within device models and parametrized by their design variables, and (ii) those that are extracted from interconnect geometry. While the former are readily available even before layout synthesis, interconnect parasitic effects may not be easily predicted.



(a) Manual analog design flow



(b) Schematic optimization analog design flow

Figure 2–3: Analog design flows

Fortunately, for technology nodes over 250nm, they are usually minor and within design margins [43]. In these cases, feasible layout solutions are usually available from top-down modeling hierarchy and schematic-optimized candidates if adequate error margins are given. Fig. 2–3(b) illustrates a circuit optimization flow based on replacing the manual schematic design procedure in Fig. 2–3(a) with the schematic optimizer.

Due to the aggressive push for high performance with existing BJT and MOSFET technologies, available error and noise margins are drastically reduced. While digital logic circuits can tolerate substantial noise in its signal, as long as the voltage level crosses the threshold voltage level within the correct time constraints, analog circuits are sensitive to signal integrity issues in general. Although not crucial for analog implementations, the need for high device density, low power and area in digital designs for high-performance

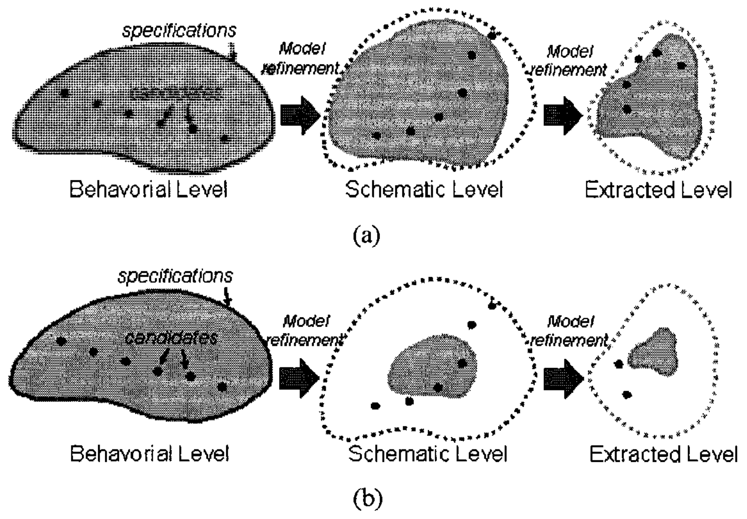


Figure 2–4: Impact of parasitic effects on design spaces (a) non-DSM, and (b) DSM design refinements

and portable applications has forced analog designs to adopt newer DSM processes for economic reasons.

With high packing density and low supply voltage, influences of parasitic effects and stochastic process variations strengthen. Minute variations of the signal can be amplified to large error, or lead to incorrect logic levels in mixed-signal systems. As supply voltages fall, path delays increase much more dramatically in a 130nm process than they do in a 250nm process [43]. Fig. 2–4 compares the conceptual impact of DSM effects from the circuit designers' point of view. Fig. 2–4(a) illustrates that model refinement throughout the design flow is gentle, and most design candidates pass the post-extraction test. The DSM environment reflects a different situation however. In Fig. 2–4(b), feasible layout solutions are not guaranteed for all candidates that are within specifications at the schematic level. This is because schematic models precede layout design, and are unable to account accurately for layout effects, while these effects are now more prominent in the DSM era. Failure of design closure will send the design process to backtrack to previous steps for costly re-starts. First-pass success rate in DSM design drops drastically. Since design automation shares the same general tool flow, all flows based on circuit-level performance measures are subject to the same impact. This issue cannot be solved by worst-case models, as performance margins are already slim. Instead, new optimization strategy is thus needed. The dominance of DSM parasitic effects have large implications affecting the implementation of future analog design automation flows.

2.4 Parasitic Effects Extraction

Electromagnetic (EM) fields governing the parasitic effects are expressed by Maxwell's equations [41]. In general, they can be solved using numerical EM field solvers. Maxwell's equations either can be expressed in their *differential* form

$$\begin{aligned}
\nabla \cdot E &= \frac{\rho}{\epsilon_0} \\
\nabla \cdot B &= 0 \\
\nabla \times E &= -\frac{\partial B}{\partial t} \\
\nabla \times B &= \frac{4\pi k}{\mu_0^2 \epsilon_0^2} J + \frac{1}{\mu_0^2 \epsilon_0^2} \frac{\partial E}{\partial t} \quad ,
\end{aligned} \tag{2.1}$$

or the *integral* form

$$\begin{aligned}\oint \vec{E}\cdot d\vec{A} &= \frac{q}{\epsilon_0}\\ \oint \vec{B}\cdot d\vec{A} &= 0 \qquad (2.2)\\ \oint \vec{E}\cdot d\vec{s} &= -\frac{\partial\Phi_B}{\partial t}\\ \oint \vec{B}\cdot d\vec{s} &= \mu_0 i + \frac{1}{\mu_0^2\epsilon_0^2}\frac{\partial}{\partial t}\int \vec{E}\cdot d\vec{A}\ .\end{aligned}$$

(2.3)

In its differential form, the entire physical design in the EM field is first discretized into a mesh that consists of small 2- or 3-dimensional tiles, and the area or volume within each tile is assumed equipotential. The potentials for the mesh at each time step are then solved. Two commonly used approaches are the *finite element* (FE) and the *finite difference* (FD) methods. They differ in whether the variables are expressed as the values of the tiles, or the differences between adjacent tiles. Both approaches effectively replace Maxwell's equations by a system of large but sparse matrix equations. The integral Maxwell's equations are a system of contour integrals of the EM fields of the physical design. Since $\oint_C \vec{F}\cdot d\vec{s} = 0$ if no sources or sinks of $\vec{F}$ are present within closed contour C, hence only sources and sinks of the fields are needed to be concerned. The *boundary element method* (BEM) employs this approach via solving *Green's function* to yield a system of small but dense matrices. In general, the FEM and FDM are preferred for large systems, while the BEM is preferred for smaller systems that need higher accuracies.

In practical IC design however, numerical field solvers are too expensive to be used repeatedly for parasitic extraction. Instead, computation results are expressed as parameters

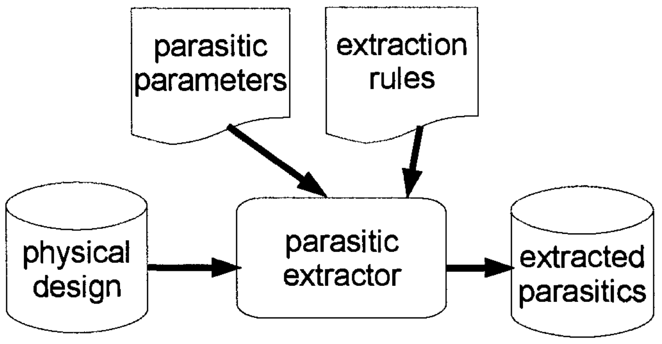


Figure 2–5: Parasitic extraction process

in terms of the material type and geometry, and are stored with other modeling parameters in the PDK technology file. To reduce modeling complexity and improve the matching of accurate parasitic models approved by fabrication foundries, the majority of physical design are realized with rectilinear geometry, except in a few special cases, such as spiral inductors, where 45 degree edges are permitted. Fig. 2–5 shows the parasitic extraction process in the IC design flow. Complex layout objects are first discretized rectilinearly according to the extraction rules. The corresponding parasitic values can then be computed by the width and length of the objects or object segments using the material parasitic properties and layer thickness parameters given in the technology file. This effectively replaces solving (2.2) for complex objects with finding, or *extracting*, the geometric parameters and substituting them into simple formulas to be given next.

Layout parasitic extraction is a crucial step in the IC design process. As a circuit design is fabricated on a silicon die, its practical performance is not only determined by the schematic devices and connections, but also dependent on the qualities and properties of the materials that made up the circuit. Extensive research on fabrication processes and associated sophisticated techniques is carried out in an effort to increase the efficiency

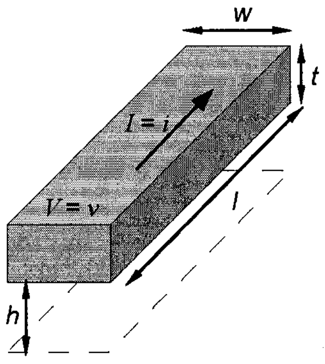


Figure 2–6: Parasitic modeling for a simple cuboid object

of the fabrication material, and to minimize the contributions of these non-ideal effects towards circuit performance. The role of parasitic extraction is to *quantitatively* predict these non-ideal effects from the layout geometry, through the insertion of passive parasitic devices such as resistors, inductors, capacitors and diodes in the extracted circuit netlist. The parasitic properties of simple shapes, such as in Fig. 2–6, can be given analytically

$$\text{inline resistance: } R = \rho\frac{l}{tw} \tag{2.4}$$

$$\text{parallel-plate capacitance: } C = \epsilon\frac{lw}{h} \tag{2.5}$$

in terms of the 2-dimensional geometry and various process-related layer thickness parameters and material resistivity ρ and permittivity ϵ. In general, most objects involved in circuit layouts have more complex geometry. Their parasitic values thus do not have closed-form formula and are solved through numerical analysis. To reduce computational and fabrication costs, most layout design rules restrict the drawn layout to contain only shapes with rectilinear edges. Through segmentation, or discretization, of these objects, their parasitic properties could be computed efficiently using a small set of physical parameters with their geometric dimensions as Z_{inl}, Z_{lat}, Z_{ver}, Z_{turns} and Z_{via} in Fig. 2–2.

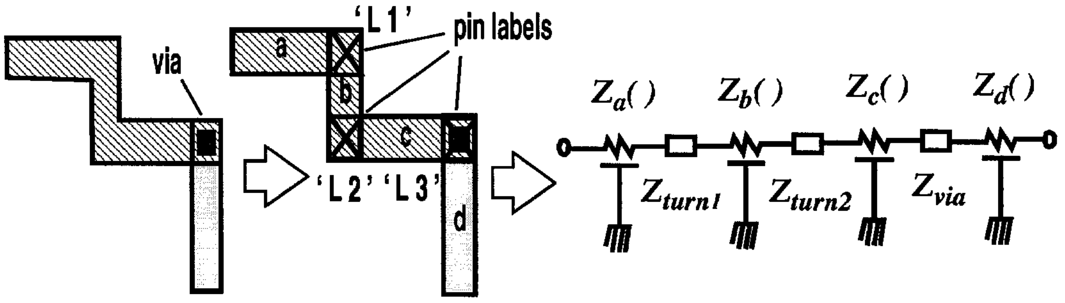
2.5 Sensitivity of Parasitic Effects due to Design Changes

Conceptually, the physical design and parasitic extraction process can be regarded as a mapping of geometric parameters to parasitic values. Except in highly controlled or the simplest cases, this mapping is practically unknown. The sensitivity analysis stems from Taylor series expansion. It offers analytical means to extrapolate complex functions based on known values and moments at certain points. The following equations define the estimation of the new parasitic value Z' with respect to new geometric parameter x', extracted parasitic value Z and known geometric parameter x based on the 1^{st}- and 2^{nd}-order sensitivity analyses respectively:

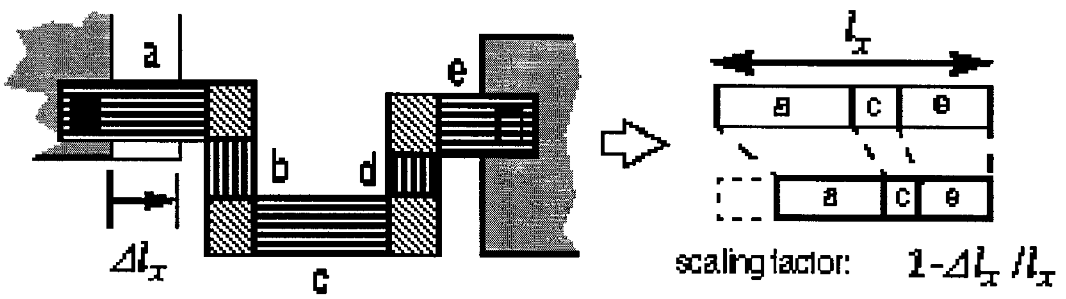
$$Z' = Z + \frac{\partial Z}{\partial x}(x' - x) \tag{2.6}$$

$$Z' = Z + \frac{\partial Z}{\partial x}(x' - x) + \frac{\partial^2 Z}{\partial x^2}(x' - x)^2 \,. \tag{2.7}$$

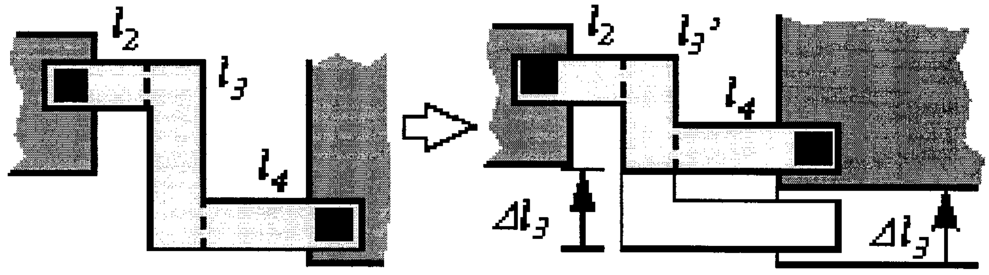
While both estimations can be used interchangeably, (2.7) requires an additional 2^{nd}-order term, but gives better estimate for higher-order functions. The 1^{st}-order sensitivity analysis is used in this work. Further investigation of the Cadence Assura parasitic extraction tool confirms the linearity of parasitic RC values with respect to wire width and length in Fig. 4–14. To derive technology-dependent resistivities and permittivities, Fig. 2–8 depicts the layout template that extracts parasitic values and sensitivity data for interconnects of various length and widths. This procedure is needed to be updated only once for each technology process. For substrate coupling, since the extraction tool used does not facilitate



(a) Labeled pin insertion at turns and vias for identification of segments in post-extraction netlist



(b) Resizing of multi-segment interconnect is applied to all segments *parallel* to the resize direction, proportional to their relative length



(c) Resizing an interconnect vertically modifies the shape and parasitics of segment l_3, while horizontal segments l_2, l_4, turns and vias remain untouched

Figure 2–7: Interconnect parasitic values update

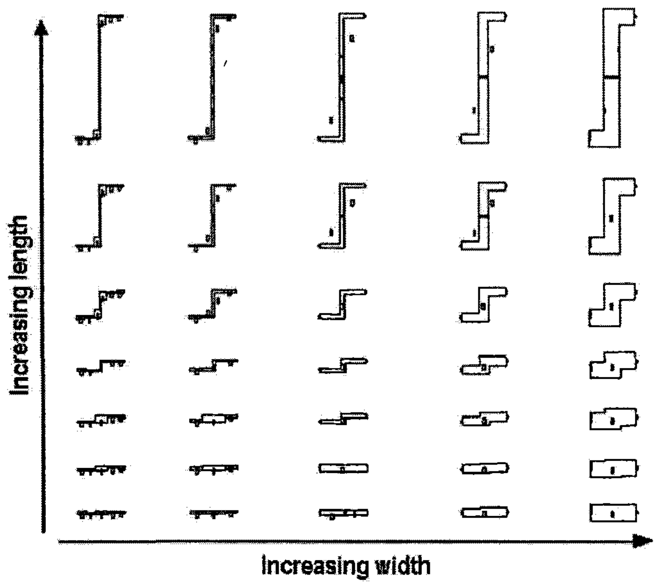


Figure 2–8: Layout template for interconnect sensitivities extraction

substrate extraction, the substrate network is derived manually from triangulation scheme. The parasitic values are computed using estimated substrate resistivity values ($10\Omega cm$ for $0.18\mu m$ CMOS process) and direct separation distances, or length of the network edges.

Parasitic models for substrate coupling, interconnect and power supply parasitics are crucial for interconnect-dominated DSM designs, as the objective functions rely on them to evaluate and rank the qualities of various placement candidates. During compaction and routing phase, both inline and lateral parasitics vary with block positions. Additionally, the vertical parasitics also vary with the interconnect lengths during the routing phase. The variations of parasitic devices depicted in Fig. 2–2 are given by:

$$Z'_{inl} = Z_{inl} + \frac{\partial Z_{inl}}{\partial l}(l' - l) + \frac{\partial Z_{inl}}{\partial w}(w' - w) \tag{2.8}$$

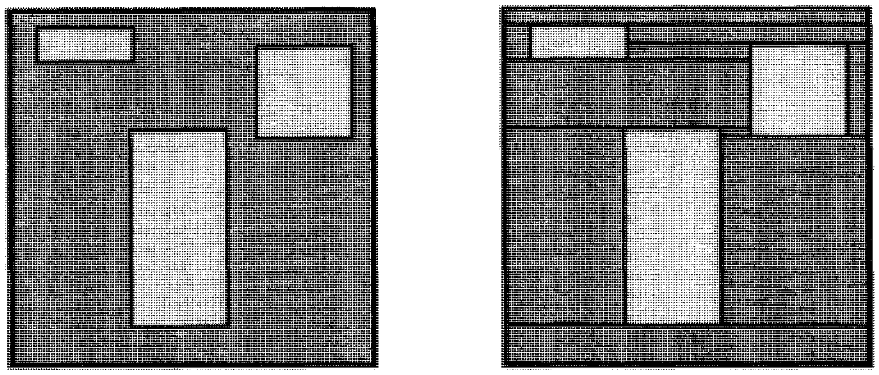
$$Z'_{lat} = Z_{lat} + \frac{\partial Z_{lat}}{\partial p}(p' - p) + \frac{\partial Z_{lat}}{\partial d}(d' - d) \tag{2.9}$$

$$Z'_{ver} = Z_{ver} + \frac{\partial Z_{ver}}{\partial A}(A' - A). \tag{2.10}$$

2.6 A Gridless Placement Encoding Scheme

Placement is a subproblem of physical design, preceded by floor-planning and followed by routing. A placement is a spatial arrangement that describes the physical position of all circuit components on the die. In the placement problem, dimensions of circuit components become fixed and represented by a set of 2-dimensional shapes. They are to be arranged in such a way that subsequent routing can be completed satisfactorily. It also requires that after routing, the circuit does not violate design rules and performs with minimal degradation. In this research, each block is rectangular and a 2-level hierarchy is used for simplicity reasons. A rectangular block can be described by two co-ordinates, such as its lower-left and upper-right corners. The 2-level hierarchy is the simplest hierarchical structure where each block represents one or more transistors and passive devices. The placement model can easily be extended to include polygonal blocks of multiple-level hierarchy.

Placement encoding schemes can generally be divided into grided and gridless types. In a grided placement, all layout objects and positions are positioned on a discrete coordinate grid. The primary advantage of it is to reduce the positional possibilities of physical design objects to a discrete number so the eventual placement and routing algorithms can be simplified and solved efficiently for complex designs.

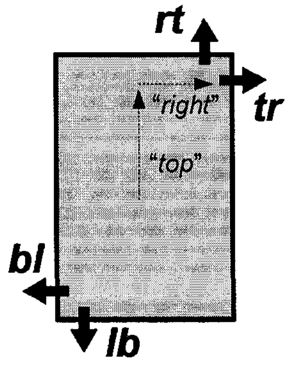


(a) A placement of 3 blocks (b) The MHS partition

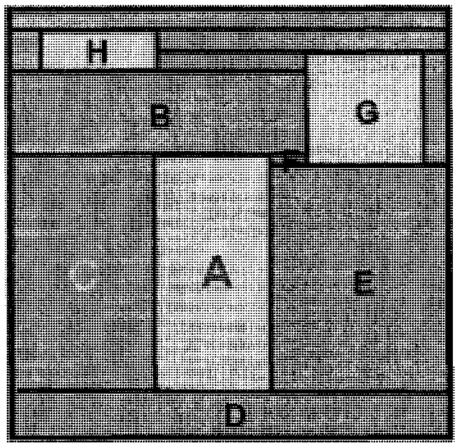
Figure 2–9: Maximum horizontal strip partitioning

For analog physical design, however, the gridless placement encoding is used since analog design are relatively small and the added flexibility is crucial for full-custom designs. In the most basic form, a placement with N rectangular blocks can be encoded by $2N$ co-ordinates. On the other hand, if the un-occupied space on the die is also encoded, operations that read and modify the placement, such as block addition, removal, plowing and compaction, finding neighboring blocks and blocks within a specified area etc., can perform much more efficiently [62]. One such encoding scheme is called *maximum horizontal strip* (MHS) [62]. This is desirable as these utilities are frequently accessed during physical design optimization and layout parasitic estimations.

In MHS, non-overlapped rectangular blocks are placed rectilinearly within an allowed rectangular boundary. Un-occupied space within this boundary is discretized into rectangles called *vacant tiles* [62]. The partition scheme is such that the width of each vacant tile is extended as wide as possible, until the boundary of blocks, or that of the placement area is reached. Fig. 2–9 shows an example of the MHS partition scheme. The same concept can also be applied vertically, yielding the *maximum vertical strip* (MVS)



(a) Corner stitch pointers



(b) Corner stitch example

Figure 2–10: Corner stitch encoding scheme

partition scheme. Depending on the block placement, either the MHS or the MVS partitions the un-occupied space with the minimum number of rectangles. MHS alone is used in placement encoding and in compaction (Section 4.5), while both MHS and MVS are used in virtual interconnect (Section 2.10.2) derivation. The *corner stitch* scheme [62] shown in Fig. 2–10(a) consists of four *neighbor pointers* for each rectangular block. Since the MHS or MVS partitioned objects are all rectangular and flush-packed together, the corner stitch pointers can be used to store the entire MHS or MVS partition in the software database. For example, in Fig. 2–10(b), the neighbor pointers for block A are: $\{rt = "B", tr = "F", lb = "D", bl = "C"\}$. For vacant tile C, the neighbor pointers are: $\{rt = "B", tr = "A", lb = "D", bl = ""\}$.

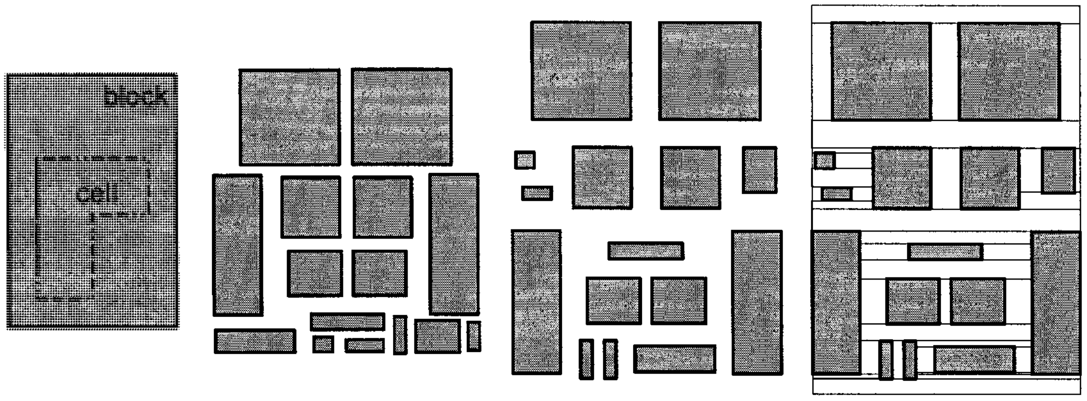


Figure 2–11: Block as cell bounding box, block packing, block placement candidates and MHS placement

2.7 Placement Candidates

Devices in the circuit are first assigned into *cells* such that all devices in each cell would be laid out together. Cells may assume any geometric shape, but since device sizes have not been determined, their boundaries are flexible, and are called *soft* blocks. For programming ease, each cell is denoted by a rectangular bounding box (or *blocks*) larger than its nominal size. Each of the terminals for external connections can be aligned with either one of the four faces of the block. Fig. 2–11 illustrates a cell and its bounding box. While a small bounding box size better reflects the actual shape of the cell, the placement would be updated if device resizing causes any cell to exceed its bounding box. After device sizes are determined, they would be replaced by *hard* blocks, the rectangular bounding boxes of the corresponding cells. Fig. 2–11 shows an example of a block, two block placement candidates and a MHS placement partition.

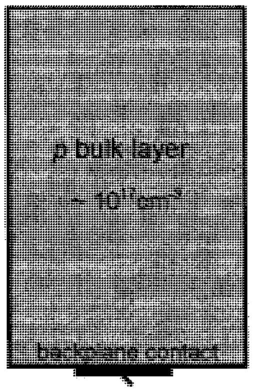
2.8 Engineering Change Order

Engineering Change Order (ECO) is defined as a formal declaration of modifications to a partially or fully completed design or product, usually applied to correct a design error or to improve performance. ECO often cannot be avoided during the design process of complex systems, due to the approximate nature of most models used. In general, the penalty of a design revision is proportional to the extent of the modification and the degree of completion. Automation tools and sophisticated data structures can decrease ECO costs to some extent. Design revisions at the behavioral or schematic level usually involve textual changes, and the penalty is very low. However, the cost of ECO rises rapidly for a physically laid out design, and becomes prohibitively highly feasible for fabricated circuits. A successful IC design flow is one that furnishes quality design models while keeping the overall ECO cost low throughout the course of the design process.

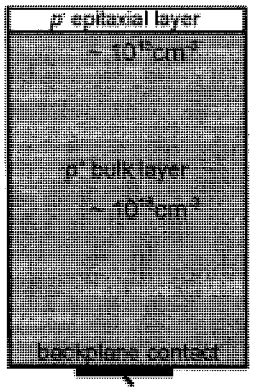
2.9 Discretization for Substrate Modeling

Substrate is a generic name of the material on the die where circuits rest on. The uniformly-doped *bulk* substrate shown in Fig. 2–12(a) is usually used for analog designs. The depth of the bulk substrate is p-doped in order to increase its insulation against the n-carriers (electrons). For *complementary metal-oxide-silicon* (CMOS) applications, the *epi-bulk* substrate shown in Fig. 2–12(b) is the most common technology. The purpose of the additional lightly-doped epitaxial layer at the substrate surface is to avoid parasitic BJTs from turning on, leading to *latch-ups*. n-carrier inhibition is further enhanced by substrate biasing at the surface and occasionally the *backplane* contact to the most-negative voltage in order to drain leakage current. The p-channel devices (pMOS) reside within an *n-well* diffusion.

Under aggressive feature size scaling, oxide and doping profile thickness are both reduced. Charge leakage to the substrate, or *substrate coupling* is common in large mixed-signal circuits implemented in sub-nanometer epi-bulk processes. Practical substrates are highly conductive and biased to the most negative voltage reference through bias contacts, guard rings, guard bands and backplane contacts, in order to drain neighboring device leakage charges [17]. Active suppression via noise-cancelling techniques [18] have also been proposed. As device density and switching activities increase, leakage current in mixed-signal DSM systems may still propagate to other devices, rather than being absorbed. Work in [57] studies the device simulation of substrate coupling between contact pairs. It concludes that the substrate behaves resistively for signal frequencies below a few GHz [57]. Further, the short distance conduction is near the surface and the bulk layer conducts better for longer distances. Coupling among nMOS devices are more common due to the sharing of the common substrate, whereas pMOS coupling are less severe since they can be more effectively isolated by isolated n-wells. More advanced fabrication processes that alleviate leakage problems and substrate coupling exist, such as *silicon-on-insulator* (SOI) and *partially-depleted* MOSFETs, but are generally more costly due to the need of new simulation models, oxide implant in the wafer manufacturing process and the associated manufacturability issues, such as tight control of silicon and oxide thickness variations [36]. The discussion continues assuming that CMOS technology implemented on epi-bulk wafers with a lightly p-doped epitaxial layer is used.



(a) Bulk substrate



(b) Epi-bulk substrate

Figure 2–12: Cross-sectional diagram of common substrate types

2.9.1 Discretization Schemes

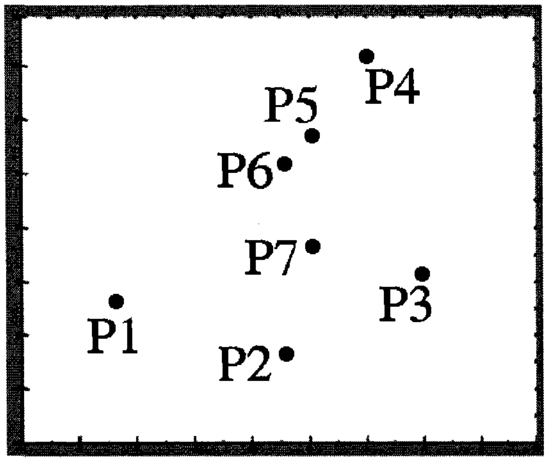
An ideal substrate is equipotential everywhere. In practice, there exist distinct substrate potentials due to the transient presence of excess charges. The distribution of the substrate potential is transient and dependent on chip activities. It is too complex to be computed analytically by systems of differential equations. Instead of solving for the continuous potential field, it is simplified to a distribution of discrete potentials via spatial discretization. The resultant network of passive devices can be integrated with the rest of the circuit schematic, and solved efficiently through circuit simulations [16]. A further model-order reduction [63] or netlist reduction [5] step can optionally be carried out. The epitaxial layer is partitioned in 2-dimensional into various tiles. Each tile carries a discrete potential at any specific time point. Any two points lying within each tile are assumed to be equipotential. The tiles can be of uniform or irregular shapes. Points of interests representing locations of circuit elements on the substrate is shown in Fig. 2–13(a). The *quad-tree* partitioning [58] and the *Voronoi* tessellation [34] schemes are illustrated in

Fig. 2–13(b) and 2–13(c). Each of the two discretization methods has its own advantages over the other. With the quad-tree partitioning, the tile shape is always a square. The regularity of the structure also constitutes easy data storage, such as assigning each square a node, and four ordered branches to its upper-left, lower-left, lower-right and upper-right quadrants. On the other hand, the Voronoi tiles size and density are optimally adapted to the given substrate points of interest. Once the partitioning is completed, a substrate network consists of parasitic resistances and capacitance is created. Substrate parasitic resistances are created between every pair of substrate interest points, if they belong to tiles adjacent to each others. The length of the shared edge (w)and the separation distance (l) between the point pairs determine the parasitic resistance $R_{sub} = \rho_{sub} l/w$. The parasitic capacitances are inserted whenever a pn junction is crossed. The values are determined by the junction capacitances.

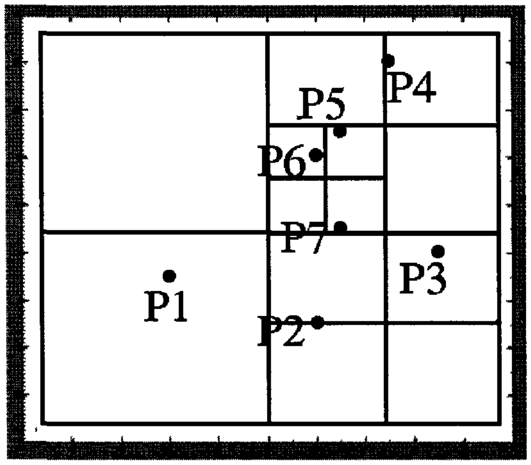
The proposed parasitic update algorithm is compatible with either partitioning approaches, and the only concern is with updating the parasitic values. The variations will be derived from the relative changes of the distances of adjacent substrate interest points, as illustrated in Fig. 2–13(d). This particular interconnect graph is obtained by Delaunay triangulation, or derived as the inverse of the Voronoi diagram from Fig. 2–13(c) at no significant computation cost.

2.9.2 Substrate Parasitic Model

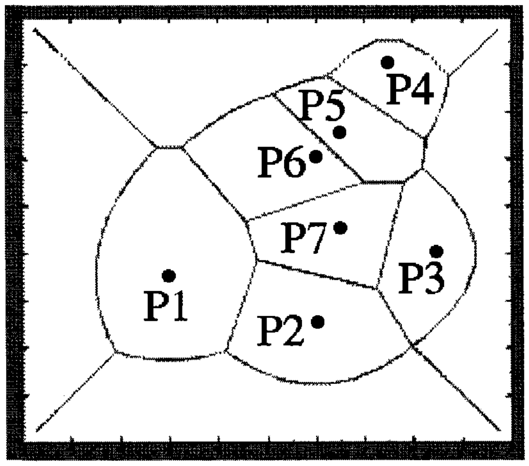
To develop a substrate model for a given circuit, the centroids of all substrate/well contacts and the channels of each transistors are used. At the placement level, the same scheme is used except the substrate contacts are replaced by block centroids to represent blocks positions. The substrate bulk is modeled by a single electrical node (the *bulk node*),



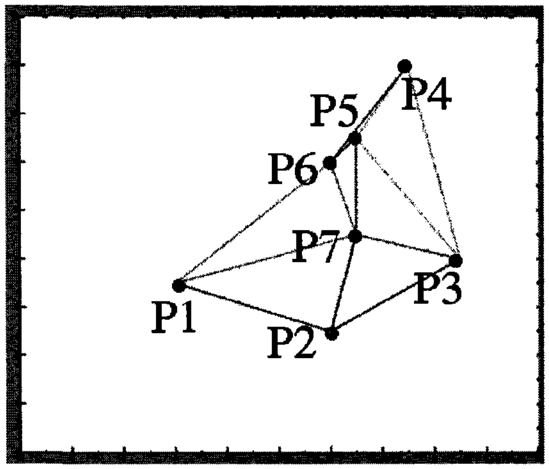
(a) Points of interest on substrate



(b) A quad-tree partition



(c) Voronoi tessellation



(d) Delaunay triangulation

Figure 2–13: Substrate discretization for parasitic modeling

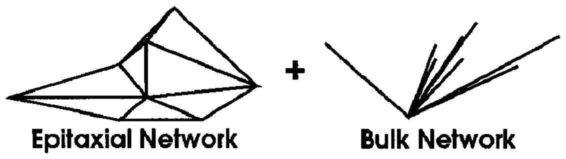


Figure 2–14: Parasitic network of the CMOS epi-bulk substrate

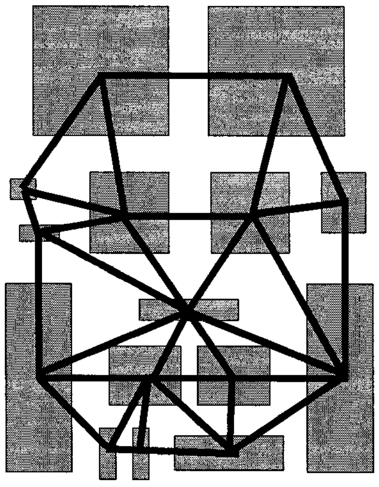


Figure 2–15: Top view of the epitaxial network (G_{12}) of the substrate model

connected to a substrate network representing the epitaxial network, as depicted in Fig. 2–14. The epitaxial parasitic network, such as one shown in Fig. 2–15, is first derived homogeneously without distinguishing between n-type and p-type substrate contacts. When the epitaxial network is connected to the bulk node, the vertical parasitic network illustrated in Fig. 2–16 is established by the R_{psub} connections. For placements or layouts involving only p-type contacts, the substrate modeling is depicted in Fig. 2–17(a). For circuits with both substrate contact types, a serial capacitance C_{well} proportional to the diffusion or well area is present to model the reverse biased diode junction. n-type contact points belonging to each well are identified. Well nodes, n-well parasitic resistances R_{nwell} are added to the corresponding *wellnodes*, as depicted in Fig. 2–16. The completed substrate model is illustrated in Fig. 2–17(b).

Conductance G_{12} in Fig. 2–17 and dashed lines in Fig. 2–16 model short-distance device-to-device or block-to-block conduction near the substrate surface that provides low

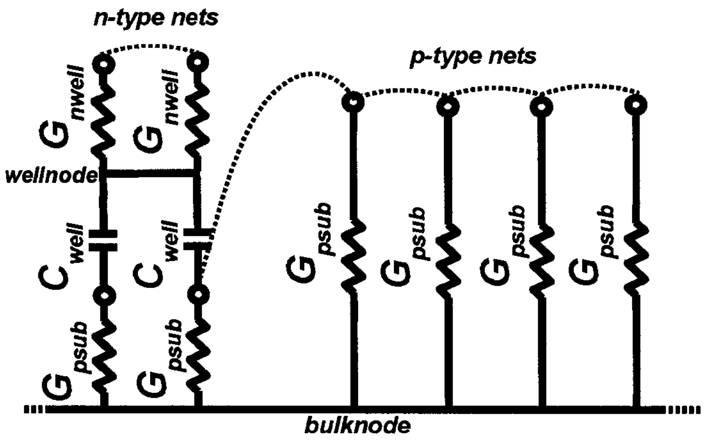
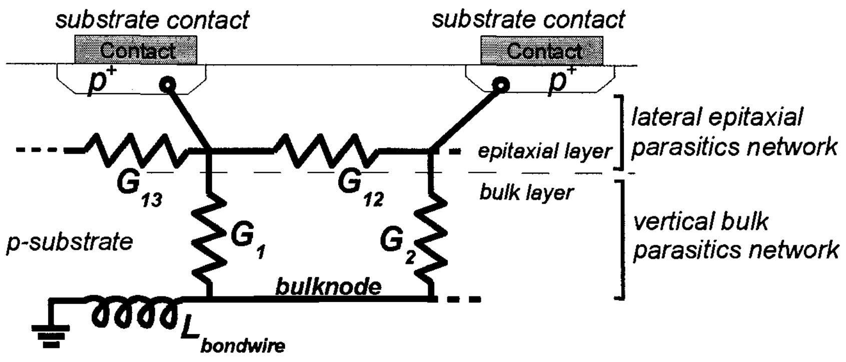
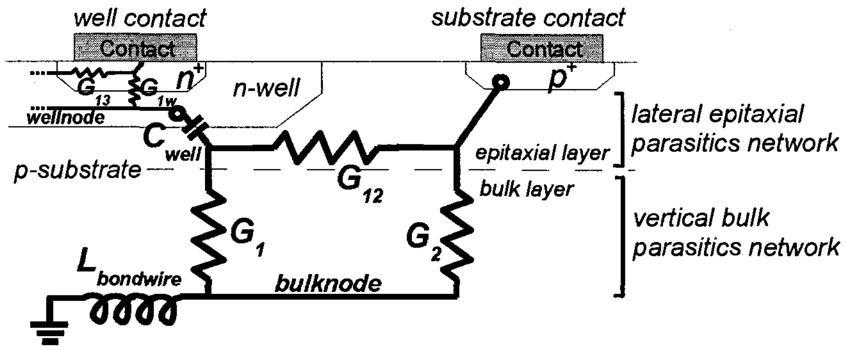


Figure 2–16: Cross-sectional view of the bulk network (G_1, G_2) of the substrate model. The dotted lines denote epitaxial network connections, among n-type, p-type and between n- and p-type substrate port nets

resistance paths for local current flow. As the port separation distance increases, G_{12} decreases and most charges conduct through G_1, G_2 in the substrate bulk. As the triangulation network connects only the neighboring devices, G_{12} among non-neighboring blocks is essentially zero. This improves computational efficiency from $O(n^2)$ to $O(n)$. $L_{bondwire}$ models the package bond wire inductance of the substrate backplane contact. If $L_{bondwire}$ is large or the backplane is left floating, global substrate coupling will significantly degrade performance. The substrate parasitics in the epitaxial network in Fig. 2–15, or equivalently, G_{12} in Fig. 2–17, varies linearly with the direct distances among neighboring devices or blocks. The vertical network of Fig. 2–16, however remains fixed, connecting every block to the common bulk node.

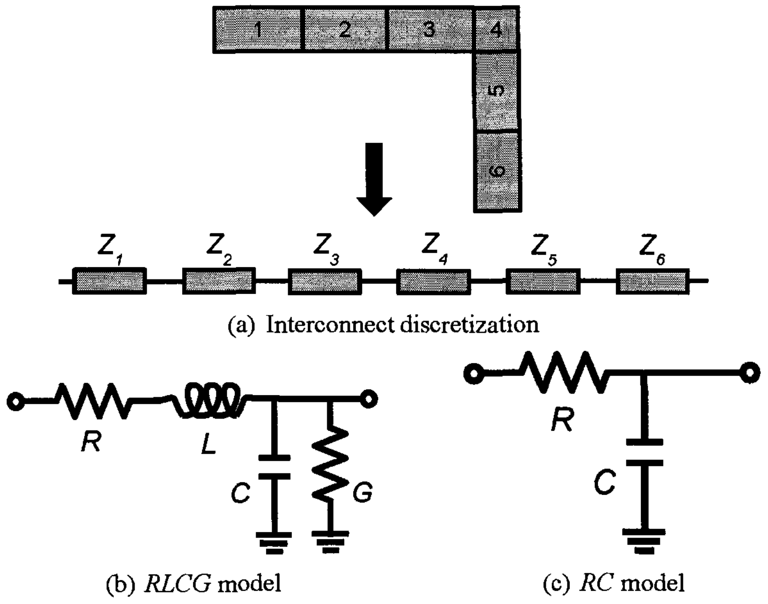


(a) With p-type contacts only



(b) With both n- and p-type contacts

Figure 2–17: Cross-sectional view of substrate parasitic models



(a) Interconnect discretization

(b) *RLCG* model (c) *RC* model

Figure 2–18: Interconnect parasitic modeling

2.10 Discretization for Interconnect Modeling

Analogous to substrate parasitic modeling problems, a practical interconnect is closely surrounded by other conductors and has its characteristic parasitic resistance, capacitance and inductance properties. When carrying current, it has a potential gradient along its length away from the driving sources, instead of being equipotential in the ideal case. Analytically or numerically solving the frequency-dependent full-wave transmission line model is impractical for CAD applications. Instead, parasitic extraction tools use the *lumped model* approach. They divide each interconnect objects into wire *segments* and 90° bend *corners*, where the wire segments and corners are represented by lumped impedance models connected serially, shown in Fig. 2–18(a). The accuracy of the lumped model can be improved by increasing the number of segments. A variety of lumped wire segment

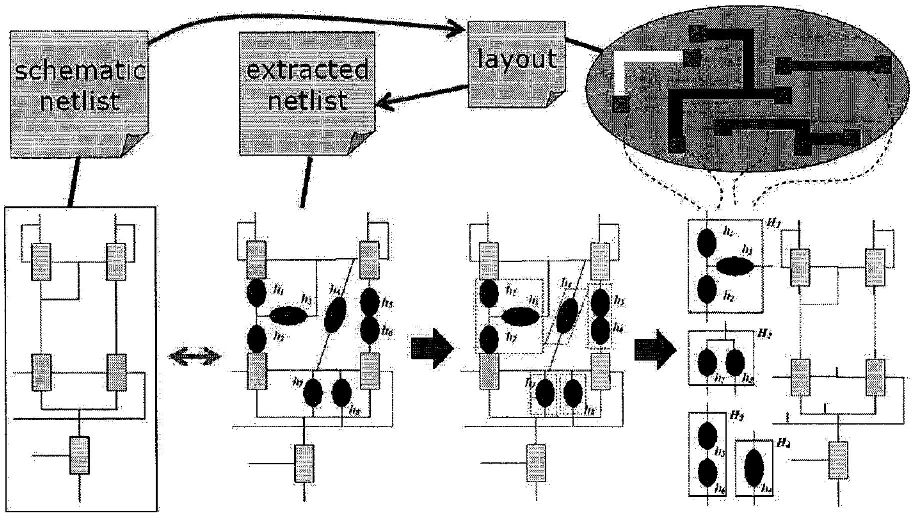


Figure 2–19: Extraction of Parasitic Macro-Devices

models, such as the *RLCG* model [61] shown in Fig. 2–18(b), can be used. The parasitic extraction routines often use a simplified lumped *RC* model (Fig. 2–18(c)) when the self-inductance L and conductance G to ground are negligible. This is the case for circuit signals that have no significant spectral power content exceeding a few GHz [57]. The extracted parasitic devices representing the physical interconnects and substrate characteristics are annotated into the schematic netlist. The resultant netlist is also known as the *extracted netlist.* Sometimes the parasitic extraction process in the EDA design flow is well-integrated with the circuit simulator, and there were no provisions to relate the physical interconnect objects with their corresponding extracted parasitic devices. In these cases, a parasitic macro-device (see next Section) recognition procedure is employed to recover these relationships by matching parasitic device groups with interconnects.

2.10.1 Parasitic Macro-Devices

Parasitic effects of layout objects may be represented by a number of passive parasitic devices that vary by their physical geometries and the extraction procedures. Due to the nature of geometric rule-based extraction programs, parasitic extraction usually results in a large number of auxiliary devices inserted into the original schematic. Their number and configuration could also change from one extraction to another. They alter the netlist topology by splitting and creating new circuit nodes. Among these parasitic devices, some influence the circuit performance more than others. However, it remains largely unknown which parasitics matter, until after many simulations took place.

A consistent extracted netlist topology is needed so that parasitic effects can be tracked. Distinct clusters of parasitic devices in the extracted netlists are grouped together into multiple-port networks called *parasitic macro-device* (PMD) H. As each PMD corresponds to a distinct interconnect in the physical design, this facilitates consistent matching of interconnects in the schematic to their respective extracted parasitics in the extracted netlist. Fig. 2–20 depicts the PMD extraction process for a circuit schematic.

Treating an extracted circuit netlist as a graph of interconnected design parameters and extracted parasitics, parasitic macro-devices are disjoint subsets of connected extracted parasitic devices in the netlist. Let $H_i(h_1, ..., h_{n_i})$ be the transfer function of the i^{th} PMD H constituted by a parasitic device $h_j \in H_i$ for $j \in \{1, ..., n_i\}$. Every H is a multi-port passive network with ports attached to schematic device ports, I/O pins, power supply or ground.

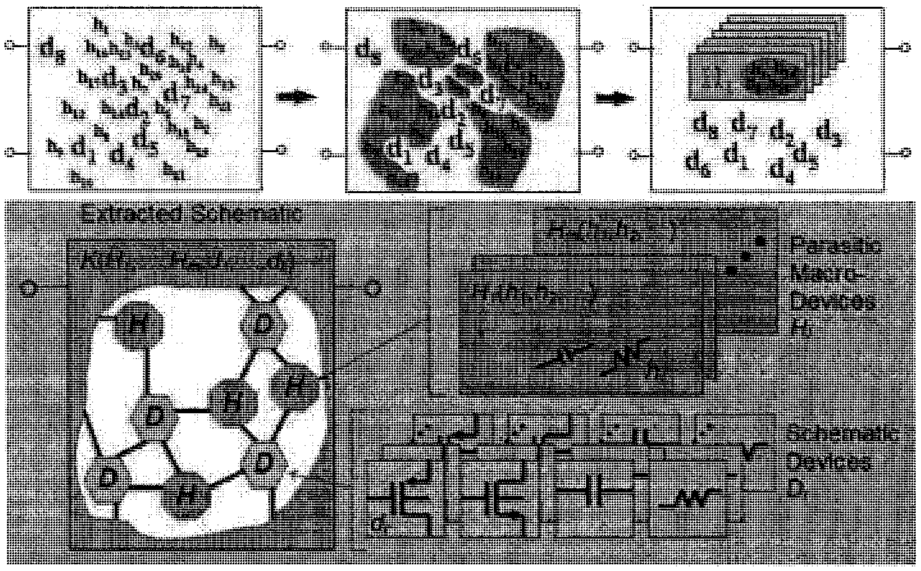


Figure 2–20: Parasitic Macro-Devices in a circuit

Algorithm in Fig. 2–21 identifies all PMDs in an extracted circuit based on a depth-first search. The algorithm initiates a search (line 4) at each port of all schematic (non-parasitic) devices, tracing all contiguous but unvisited parasitic branches. There are no redundant searches since every parasitic device belongs to a unique parasitic macro-device. Since every PMD is a connected graph made up purely of parasitic devices, each search yields at most one PMD. After all PMDs are identified, the parasitic values are matched with the corresponding linear interconnect segments, delimited by pin labels. In the parasitic modeling algorithm, H is represented by *modified nodal formulation matrices* [68]. The parasitic models keep track of how each adjacent H impacts performance, by the performance sensitivity, when design parameters change. Performance sensitivity of each H can be computed efficiently using its *adjoint network* H_a [68].

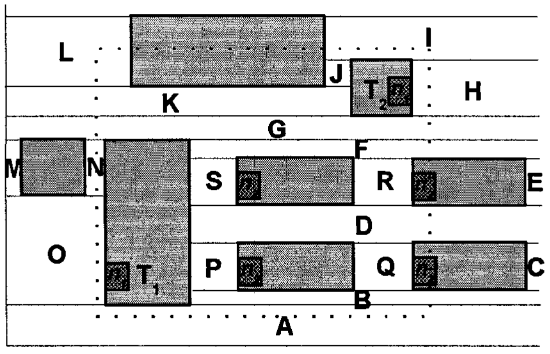
```
procedure **find_PMDs**:
//Input:  ext_ckt_graph
//Output:  PMDs
foreach sch_dev in sch_devs:
 foreach port in sch_dev:
  net = unvisited_net_connected_to_port(port)
  PMD = explore_connected_par_dev(net)
  append PMD to PMDs
return PMDs

 function **explore_connected_par_dev(net)**:
  devs = unvisited_devs_connected_to_net(net):
   foreach dev in devs:
   if dev is parasitic and unvisited
    save par_dev in PMD
    net = unvisited_net_connected_to_dev(dev)
    explore_connected_par_dev(net)
 return PMD
```

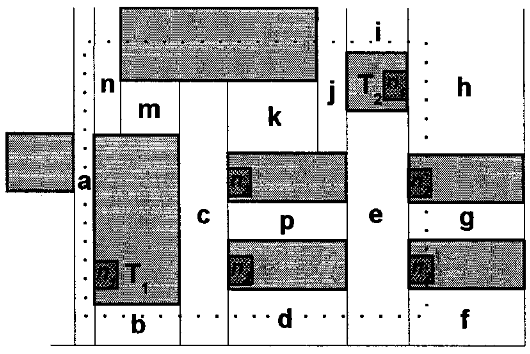
Figure 2–21: Algorithm: Parasitic macro-device identification

2.10.2 Virtual Interconnect Model

When a circuit design is yet to be finalized, its physical design is incomplete while the device blocks remain *soft*. Parasitic extraction requires the floor-planning, placement and routing steps to be completed. After studying parasitic extraction results of various physical designs sharing an identical schematic, it is concluded that the layout parasitic effects varied widely with various placements. The variations become much less among layouts with different routing configurations, but sharing the same placement. In particular, the interconnect parasitics become tractable with the length of the interconnects, within certain limits. In order to realistically but efficiently estimate the layout effects without undergoing the computationally intensive physical design steps, it first requires the placement



(a) MHS partition



(b) MVS partition

Figure 2–22: Terminals T_1, T_2 to be connected, and the geometric partitions

configuration to be constructed, and then compute the *virtual interconnects* (VIs) among terminals to be connected. Based on the VIs the interconnect parasitics are found.

In formulating the construction of the VIs, common practices of analog designers are observed:

- Interconnects should take the most direct path that do not route over cells.
- Changes in metal layers are discouraged unless necessary.
- The number of turns per interconnect is minimized.

The following implementation computes VIs that preserve these characteristics for a given placement. Fig. 2–22 shows the MHS and MVS partitions near terminals T_1 and T_2. The MHS and MVS partitions are fixed throughout top-level interconnect estimations, and are reused across schematic optimization as long as resized cells do not exceed their block boundaries. Firstly, *vacant* tiles within the bounding box of the terminal pairs to be connected are fetched from the MHS and MVS partitions of the placement. The VI would be entirely enclosed within this bounding box, indicated by the dotted rectangles in the figures. The adjacency graphs among *vacant* tiles (Oa, Aa, Ab, Ac, Ad, Ae, Be,

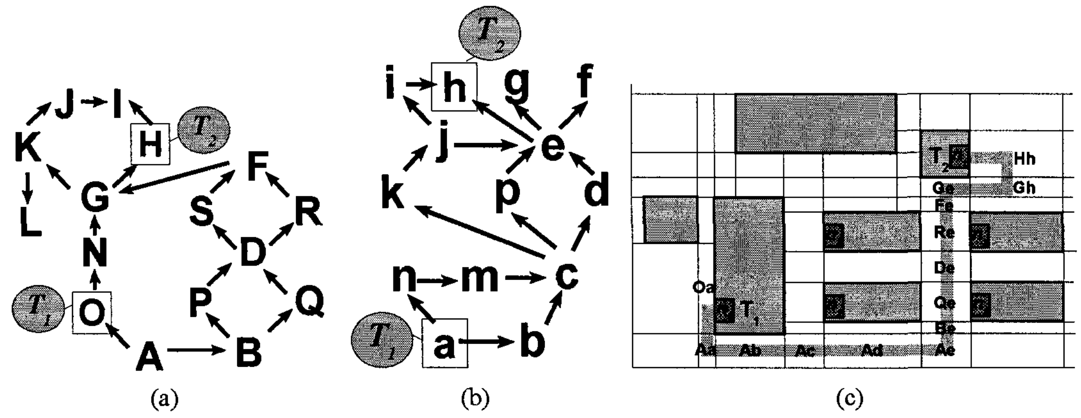


Figure 2–23: Vacant tiles adjacency graphs for (a) MHS partition, and (b) MVS partition. (c) Virtual interconnect

$Qe, De, Re, Fe, Ge, Gh, Hh$) between T_1 and T_2 are shown in Fig. 2–23(a) and 2–23(b). The vertices of the graphs stand for the vacant tiles, and the directed edges denote the traversal between two adjacent vacant tiles U and V. An associated traversal cost $e(U, V)$ is given by the combined lengths of U and V. The lengths are their widths or heights that are parallel to the direction of travel. $u \rightarrow v$ denotes block v *on top of* u for the MHS graph, and v *on right of* u for the MVS graph.

Fig. 2–24 shows the double graph traversal algorithm that derives virtual interconnects between two terminals. Referring to Fig. 2–22 again, the virtual routing always begins at the lower-left of the terminal pair (line 3) in the placement. One advancement can be made on either one of the MHS and MVS graphs in Fig. 2–23(a) and 2–23(a) at one time. To minimize the number of turns, the advancements following the lowest cost options are made on the same graph as long as the block pairs in the MHS and MVS graphs intersect. If no such option is available (line 9), then advancement is made on the other graph. Switching between the graphs also indicates that a turn is being made. The

```
IntconnEstimate(terminals,MHS,MVS)
foreach (T1,T2) in terminal pairs
 determine bBox, positions of (T1,T2)
 form adjacent graphs AGH,AGV from MHS,MVS,bBox
 Dijkstra traversal of AGH and AGV at T1
 while T2 not reached
  if blks in AGH,AGV intersect then
   proceed and accumulate length costs
  else
   toggle graph advancement
   add turn count
  end if
 end while
end foreach
```

Figure 2–24: Algorithm: Route finding between terminals T_1 and T_2.

algorithm continues until the destination terminal is reached. Example of a virtual routing between terminals T_1 and T_2 is depicted in Fig. 2–23(c).

For multiple-terminal nets, parasitics of the dominant conduction path between two given terminals are varied with geometric changes, while the rest of the parasitics remain fixed. The dominant conduction path is the least resistive path through the net connecting the given terminal pair. The interconnect polygon is geometrically partitioned into vertical, horizontal and corner tiles, as depicted in Fig. 2–25(a). The tile pieces are represented by a connectivity graph shown in Fig. 2–25(b), with the vertices and edges representing the vertical or horizontal pieces and the corner pieces respectively. The associated weights are equal to their inline parasitic resistances. Derivation of the 2-port interconnect is based on the *Dijkstra's* shortest path algorithm to efficiently solve for the least resistive path Y between every terminal pairs to be connect. Application of the algorithm is described in detail in Section 2.10.3. Effectively, a subset of the interconnect links is distinguished between every two terminals. Fig. 2–25(c) traces the least resistive paths from T_B to all other terminals. The 2-terminal VI derivation described above in Fig. 2–24 can then follows.

Fig. 2–26(a) shows the interconnect model consisting of devices representing the parasitic resistances and capacitances of the connection. Fig. 2–26(a) the interconnect model for Y. Using parasitic parameters derived in Section 2.4 and an assigned interconnect width w, the parameter values can be obtained as follow:

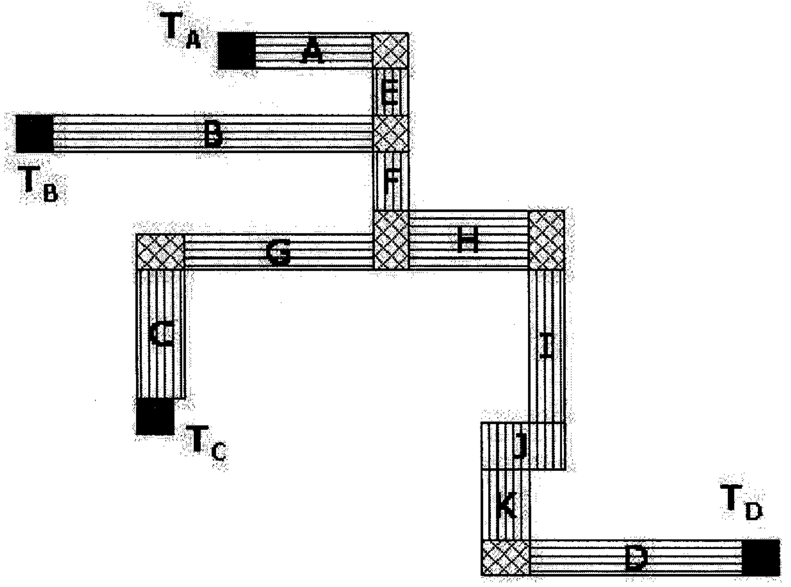
$$R_{series} = l\frac{\partial \mathcal{R}_{inl}}{\partial l} \tag{2.11}$$

$$C_{ver} = lw\frac{\partial \mathcal{C}_{ver}}{\partial l} \tag{2.12}$$

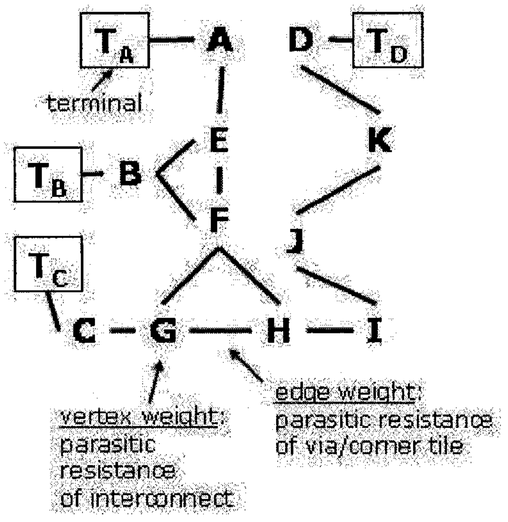
$$R_{turn} = n_{turn}\mathcal{R}_{turn} \tag{2.13}$$

$$R_{via} = n_{layer}n_{via}\mathcal{R}_{via} \tag{2.14}$$

Here, R_{series} and C_{ver} are determined by the length of Y, and R_{turn} is determined by the number of turns made in Y. R_{via} is determined by n_{via} the number of vias in a via group, and n_{layer} the total number of via layers between the metal layers of the terminal pairs. Fig. 2–26(b) further illustrates the sample solution from Fig. 2–23(c). It has 6 turns and 1 via layer between its terminals residing on *metal 1* and *metal 1* layers. Its width is user-defined. Its length is equal to the sum of the heights and widths of the blocks it traversed in the MHS and MVS partitions ($OABQDRFGH$ and $abcdeh$) respectively, minus a fractional portion of the *head* and *tail* blocks ($OHah$).

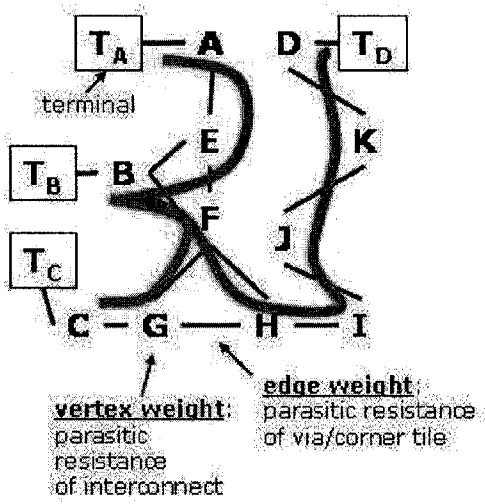


(a) Discretization of multiple terminal interconnect



T_A
A
D
T_D
terminal
E
K
T_B
B
F
J
T_C
C
G
H
I
vertex weight: parasitic resistance of interconnect
edge weight: parasitic resistance of via/corner tile

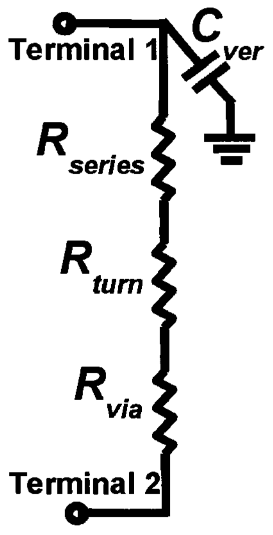
(b) Connectivity graph



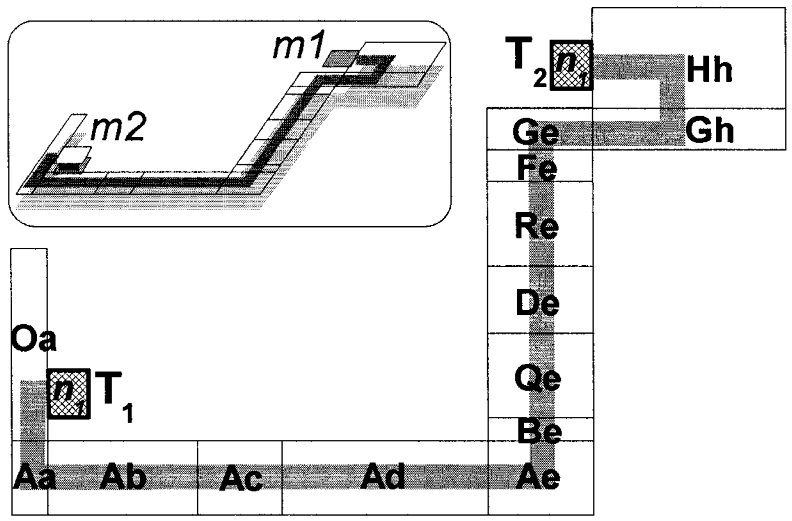
T_A
A
D
T_D
terminal
E
K
T_B
B
F
J
T_C
C
G
H
I
vertex weight: parasitic resistance of interconnect
edge weight: parasitic resistance of via/corner tile

(c) Shortest paths to terminal T_B

Figure 2–25: Finding the shortest paths within a multiple terminal interconnect



(a)



(b)

Figure 2–26: (a) Virtual interconnect model (b) Interconnect example

2.10.3 Adaptation of Dijkstra's Shortest Path Algorithm

The Dijkstra's algorithm given in Fig. 2–27 is employed to search for the least resistive conduction path within a multiple-terminal interconnect between a terminal pair. It is an exhaustive *breath-first* search algorithm that finds the *closest* vertex of a given vertex in a connectivity graph. The graph is undirected, stored as an adjacency list, and has non-negative costs assigned to all its vertices and edges. Beginning at a given initial vertex at line 8 in Fig. 2–27, it compares the current costs of all its immediate neighbors and the costs connecting to them (line 12). Of those that the new costs are lowered, the `previous` pointers (line 13) are re-assigned, and the accumulated costs, or distances are updated (line 14). The vertices traversal repeats again an unvisited vertex with the lowest accumulated cost (line 8) until all vertices are visited. The shortest paths from each vertex to the given initial vertex can be traced by following the `previous` pointers.

```
dijkstra(G(V(weight),E(weight)),init_vrtx)
visited_vrtx = empty
unvisited_vrtx = V
dist(init_vrtx) = 0
dist(unvisited_vrtx) = infinity
prev(init_vrtx) = nil
while unvisited_vrtx not empty
 u = min(dist(v)) for all v in unvisited_vrtx
 remove u from unvisited_vrtx
 insert u into visited_vrtx
 foreach v connected directly to u
  if dist(u) + weight(u,v) + weight(v) < dist(v) then
   prev(v) = u
   dist(v) = dist(u) + weight(u,v) + weight
  end if
 end foreach
end while
```

Figure 2–27: Algorithm: Dijkstra's shortest path algorithm

CHAPTER 3

ANALOG DESIGN AUTOMATION FRAMEWORKS

Computer-aided design (CAD) tools have been in existence since the early days of computing. While the bulk of effort in CAD has focused on digital design, some of the earliest CAD research focused on the analog design problem [28]. Although most of CAD research is dedicated to digital systems, the recent re-emergence of analog circuitry as part of a mixed-signal system or system-on-chip and the shortened design cycle has renewed the interests in analog design automation.

Traditional *computer-aided design* (CAD) tools enhance designer productivity by simplifying design procedures and automating mundane and tasks. Procedures related to design information input and task executions are regarded as *front-end* components, such as the input interfaces for the schematic and layout design, the analysis specifications and the test bench setup for performance evaluations. The *back-end* components operate on the design data according to the technology in use. They include design rules checking, noise analysis, layout and schematic matching, parasitic effects extraction, netlist extraction, circuit simulation layout placement and routing, etc. These are stand-alone *point tools* that users invoke sequentially. when the `time-to-market` of modern complex systems is shortened, these tools must be well-integrated to become a tool suite. The key to such integration is the automated decision making with minimal or no user interventions. An efficient framework promoting efficient transfer of design information between the front- and back-end components is also vital.

3.1 Circuit Design Automation

Analog synthesis involves selecting the circuit topology and circuit parameters to satisfy the given performance specifications. Creating circuit topologies from scratch requires a combination of strong concept of circuit knowledges and creativity [28]. It is highly-challenging to automate it in software and so far largely unsuccessful in practice. The design space has too much details and too many degrees of freedom to be understood by a software model, especially when execution time and other resources are limited. Published implementations are either confined to a narrow set of applications or involve expert rules that were circuit-specific and difficult to construct. On the other hand, topological *selection* based on evolutionary algorithms [45] that select the best candidate with respect to the given specifications among many pre-constructed topologies is a viable alternative. As a result, most practical analog circuit topologies are constructed manually.

After the circuit topology is determined, selection of circuit parameters takes place. The problem at this stage is relatively less complex than topological synthesis. The circuit topology can be seen as an unknown function, mapping circuit parameters to system performance measures. A typical analog circuit has from tens to hundreds of circuit parameters and a few tens of performance measures. From the automation point of view, the problem can be casted as an optimization problem across the required performance measures. Numerous contributions to automate this step have been made. They can be divided into three types:

- **Template-Based Optimization**

  Circuit design is assembled from elements from a library of *soft* templates or predesigned circuit building blocks with pre-defined performance descriptions. This approach attempts to reduce the complexity of the design problem from the number of design variables to approximately the number of circuit building blocks. An important benefit is that post-schematic layout parasitic information is available up-front during circuit design. On the other hand, the approach is versatile only for semicustom designs. It is rare that high-performance design using specific constructs can be fully implemented with elements from the templates or libraries offered by the same vendor. Soft templates partially alleviate this by providing certain degree of flexibility. Another important concern is the maintenance cost due to technology-dependent nature of the library implementation.

- **Constraint-Based Synthesis**

  The ranges of the design variables concerned are refined by agroup of equations that describe the specificities and constraints of the circuit. A search mechanism chooses and varies the design variables iteratively based on the evaluated performance. The advantage of this method is that it is only mildly technology-dependent and performance evaluation is quick. Hence it is also called *design space exploration* due to the ease of examining the performance measures of wide ranges of design parameters. However, the drafted equations are application-specific and requires circuit knowledge. A special case is when both the system of performance and constraints functions are *convex*, or posynomial, functions, then the *globally* optimal performance can be solved efficiently by *geometric programming* [56], even for practical high-dimensional problems. Nonetheless, equations are necessarily approximated analytical models, which do not provide adequate accuracy for DSM circuit optimization [55].

- **Simulation-Based Optimization**

  Instances of circuit designs in various topologies and device sizes are represented by netlists. Through the use of circuit simulators, performance of each design candidate is obtained. Parallel processing techniques and batch processing modes can be used to speed up the simulation processes. A search mechanism chooses and varies the design variables iteratively based on the simulated performance. This method takes advantage of recently available high-performance processors to obtain accurate circuit evaluations through simulation. While no circuit modeling effort is needed, its effectiveness is primarily limited by the simulator efficiency for large circuits. In cases where a particular type of circuit is unsuitable for direct simulations (e.g. Phase-locked loops), a pre-simulation modeling step is performed.

A number of analog circuit design automation applications using a combination of the methods outlined above are reviewed below.

3.1.1 Equation-Based Design Automation Methodologies

Numerous works have been published in the application of convex programming to the synthesise or optimization of specific analog circuits with respect to a set of assigned specifications. In each of these works, a number of design constraints and performance objective equations unique to each design are drafted prior to the design process. These constraint and objective functions are derived from designers' knowledge and reduced-order models specific to circuit functionalities and topologies for each design.

Geometric programming is applied in [26] and [44] to several analog circuit design problems, such as amplifiers and Op-Amps. A few design constraints such as gain, bandwidths, slew rate, power supply rejection ratio, bias current and output resistance relationships are expressed in terms of design variables. Since the feasible design space is not a convex polygon in general, an approximation of the feasible set is created by applying more restrictive constraints to the existing design constraints such that the resultant constrained design space becomes convex. This is called the *interior point method*, and approach the speed of linear programming solvers with current implementations [13].

Circuit modules and building blocks are alternative ways to implicitly describe design parameter relations, instead of drafting equations and constraints. At the early days when MOS transistors were just adopted by analog designs, [31] envisioned the need for analog CAD, and proposed a topology selection flow based on circuit building blocks with parametrical adjustment possibilities. In [69], a figure-of-merit is defined for a number of circuit topologies or architectural elements for analog-to-digital converters. During architectural selection, the variations of the corresponding figure of merits together with other design parameters are observed within the design space to realize an optimal design.

In an effort to provide improved flexibilities to circuit templates and building blocks, the CAIRO+ project [42] offers an analog circuit description language that facilitates management of analog electrical and geometrical constraints. In [3], [27], circuit design knowledge, and performance equations of a sigma-delta modulator design flow are encoded in CAIRO+ functions. A combination of equation- and simulation-based techniques are employed in combination with layout generation functions, modification functions of specifications and device sizes. An evolutionary algorithm is used in [45] to perform un-supervised selection of "thousands" of circuit topologies with respect to the given functionalities.

3.1.2 Simulation-Based Design Automation Methodologies

Although equation-based methodologies are efficient in arriving at design solutions, they do so by trading-off accuracy for the speed of simplified equations and low-order models or via semi-custom building blocks. As a result, design closure and optimality are not guaranteed when candidates fail to converge at the verification stage. Besides, the amount of automation is reduced when specific expert knowledge is needed for each circuit type. The rationale for simulation-based design automation techniques is to improve synthesis reliability and eliminate the difficulties in obtaining analytical descriptions by *simulator-in-the-loop* optimization.

The notion of eliminating convex circuit constraints implies that the conditions for geometric programming are no longer valid. It is replaced by empirical search algorithms such as *genetic algorithms*, *simulated annealing* [62] etc., to overcome the issues of local minima often met on non-convex and unsmooth cost surfaces. ASTRX/OBLX [49] has a simulated annealing-based optimization architecture. The ASTRX first compiles circuit parameters and topologies into performance cost functions. The circuit parameters are then perturbed and settled within the OBLX simulated annealing algorithm to produce and optimized solution. A genetic algorithm is coupled with circuit simulation in [65] to evaluate and guide the optimization analog circuit device sizings. Phelps et al. proposed the *Anaconda* tool [51] that combines simulation-based optimization with evolutionary pattern search techniques spanning multiple processors to cope with high volume simulator access.

3.2 Modeling Parasitic Effects in the DSM Performance Space

To perform layout-aware circuit optimization, means to extract or estimate the parasitic effects for each given circuit design are needed. The modeling process is confronted with the diverse number of physical solutions. For a given circuit, a number of placement configurations are possible. Distinct routing solutions can be found for each layout placement configuration. The parasitic characteristics of each of these layouts are therefore different. Hence, the post-extraction performance of a single circuit design is spanned by a range of performance measures.

Two reasonable approaches to modeling the parasitic effects are to either: (i) perform optimization at each stage of the physical design process, or (ii) select one *representative* layout solution for each circuit design, out of the large feasible layout set. The former option traverses layout options for each circuit design, while the latter option compares among circuit designs together with their layout parasitic effects. Although it can be theoretically deduced that some layout designs perform better than others, it becomes expensive to explicitly implement more than a few of the set of physical designs. Most layout-aware methodologies opt for the latter case, for efficiency reasons. For fair comparisons in the latter case, the layout design used ought to be the most representative among the many possible physical design solutions for a given schematic. It is also crucial to invest the limited design efforts on promising candidates. In the proposed flow of this work, each layout synthesis is preceded by several iterations in which parasitic estimation models are used. The parasitic models are based on actual placement and layout schemes that anticipate the habits of typical analog layout designers.

The rest of this chapter first investigates the impact of the circuit performance space and design automation due to the migration towards deep-submicron and nanometer technology processes. A number of possible approaches are then discussed. It is followed by a review of existing layout-aware design automation methodologies. The methodology proposed by this work will be given in the next chapter.

The performance space of a hypothetical design automation problem is shown in Fig. 3–1. The white, grey and black circles denote a chosen schematic topology through schematic and layout syntheses. From a fixed circuit topology, several schematic designs are synthesized. With respect to these schematic candidates, a number of layout solutions are created and scattered across the 1-dimensional performance space is illustrated in Fig. 3–1. The layout performance degradation effects are divided qualitatively into 4 types. Fig. 3–1(a) depicts situations where layout candidates which share a common schematic candidate have narrow performance variations. It implies that either it is a conservative design, or robust layout techniques and process technologies have been used. In this type of scenarios, the conventional design flows and models work well and ensures

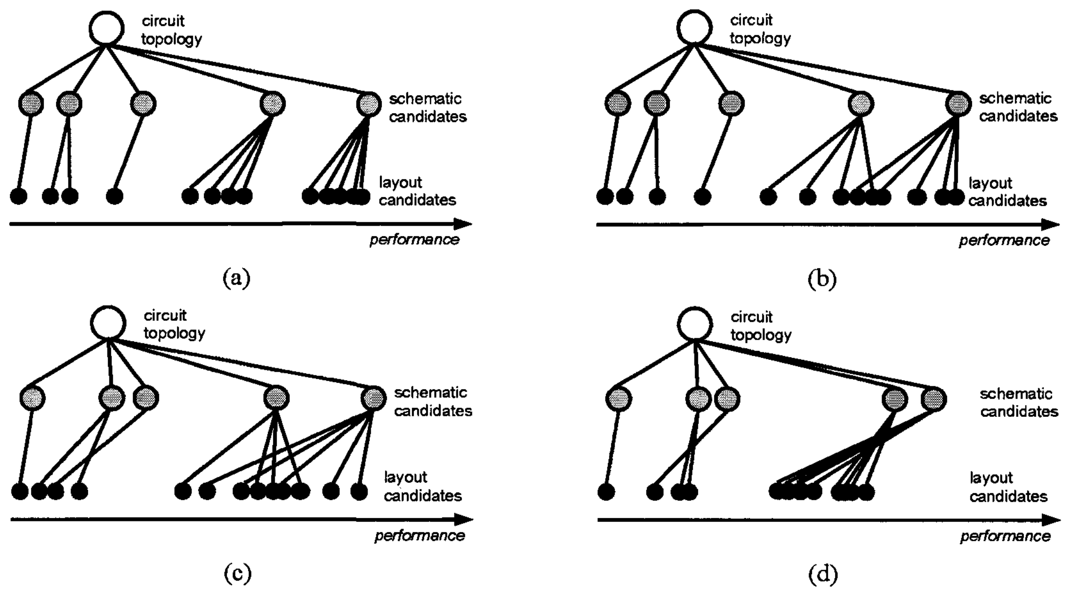


Figure 3–1: Scenarios of design candidate performance (a) Type I: Small fabrication effects and small layout spreads, (b) Type II: Small fabrication effects and large layout spreads, (c) Type III: Large fabrication effects and large layout spreads, and (d) Type IV: Large fabrication effects and small layout spreads.

high closure probability. Note that the performance degradations of the layout candidates can be attributed to differences in placement configurations and layout techniques.

In Fig. 3–1(b), the circuit performance is more sensitive to the effects of fabrication, and the performance spread is no longer compact. There are cases where performance figures of layout candidates cross-over each others, indicating that a lesser performer in the schematic level may yield a better performing design at the post-extraction stage. Although not the optimum design at the schematic level, it is more robust against layout effects. The higher occurrence of this type of inversion, the more crucial physical design optimization is to ensure high closure probability. Nonetheless, in these Type II circuits, traditional schematic design flows with few layout synthesis can still provide feasible solutions within several ECO iterations.

Migration to smaller fabrication process nodes results in post-extracted performance being even more sensitive to fabrication effects, and will look similar to the performance of Type III circuits in Fig. 3–1(c). The globally optimum layout candidate does not belong to the top-performing schematic candidate. This case is typical with high-performance analog and RF designs, in which the optimum layout could not be easily obtained without evaluating several layouts for many schematic candidates. Traditional design flow may not converge to a feasible solution efficiently.

Note that layout techniques and fabrication effects are independent issues. By designing the schematic candidate to be less sensitive to layout design variations as shown in Fig. 3–1(d), Type IV circuits, where the performance spread is compact despite the fabrication effect being strong. This is the case for some small RF circuits, when possible layout configurations are limited. Performance figures of different layout designs of the same schematic are similar, and can be approximately found by generating one layout instance. Thus, if layout design variations can be minimized, the near-optimum layout can be almost certainly found by synthesizing and comparing exactly one layout instance for each schematic candidate in the set of top schematic candidates. Because the practical design space is complex, it is often difficult to visualize the performance space beforehand. Existing layout performance evaluation is a verification process. The slow layout synthesis step is a bottle-neck that needed to be improved for optimization algorithms to run efficiently. The following alternative approaches are studied:

1. Enforcement of strict design margins and layout design robustness in schematic design
    - Conservative performance expectation
    - Schematic candidates robust against layout design variations (Type IV as in Fig. 3–1)
2. Improvement of layout synthesis and parasitics extraction throughput
    - Layout cell libraries
    - Parametric layout templates
3. Substitution of layout synthesis by parasitic estimations
    - Re-define the notion of schematic model to convey layout topology
    - Extrapolation: Local incremental layout parasitic approximations

Strategies that implement these approaches are summarized in the following sections. The goal here is to solve Type IV circuit optimization problems efficiently, while using heuristics for Type III problems.

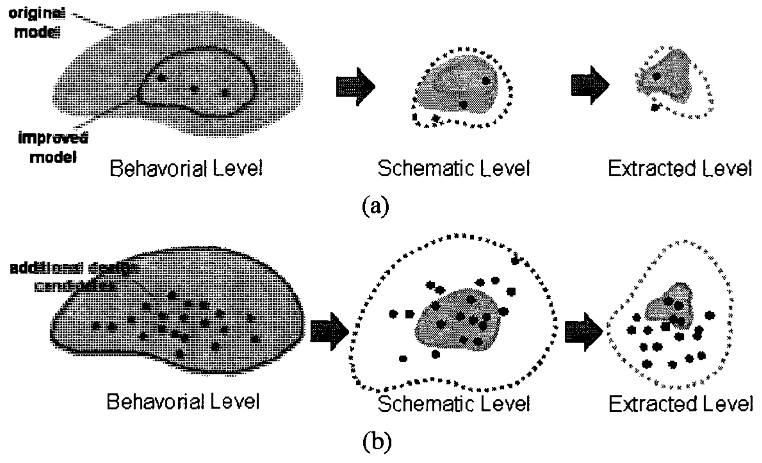


Figure 3–2: Remedies of non-linear DSM effects on design spaces (a) more accurate models reflecting DSM effects, and (b) better design space coverage

Adding to the modeling complexity, certain specific objectives are inter-related and crucial to the particular types of circuit. For example, matching the power supply and ground grid spacings, or row heights of standard-cell library for any matching cells, and minimize area consumption for digital circuits. For analog circuits, matched device pairs may require inter-digitated configurations or symmetrical placement along the median, while leaving adequate space for interconnections and noise shielding. Symmetry and robustness against process variations are the key objectives. For an RF circuit, the area-dominant inductors and capacitors are carefully laid out and placed, dominating the total layout area. Extra empty space are given among components to aid noise shielding. Extensive and accurate parasitic simulations are mandatory.

Evaluating schematic performance is relatively inexpensive. Performance of many design assertions can be verified and compared quickly, and the best option could be obtained in relatively short amount of time. Hence, algorithms such as hill-climbers, simulated annealing and genetic algorithms are effective in schematic optimization. On the other hand, existing layout generation process and extraction needed for simulations are much more expensive. Typical layout design space is non-flat. Feasible layout solutions are overly complex to be modeled analytically, and the number of variations is too large to cover exhaustively. Therefore, performance-driven layout optimization is not practical without efficient means to speed up the schematic to extracted netlist (netlist with layout parasitics) conversion process.

3.3 Existing Automation Approaches with Provisions for Parasitic Effects

Building an approximate performance space at the schematic level has been of great interest for providing simple circuit models with architectural constraints that can fully enable top-down design methodologies. A number of proposed design automation approaches described above either efficiently explore a large number of design possibilities using circuit simulators, or methods to describe the design or solution space by design constraints and performance equations in order to obtain the optimum solution without simulation needs. In general, they are carried out without notions of the underlying layout feasibility and post-extraction performance tradeoffs.

Recently, there is new momentum to improve on these design automation flows with regard to parasitic effects due to deep-submicron design closure issues. Template-based methodologies are among the least challenging approach to annotate parasitic information. Several studies proposed the use of templates, such that extracted layout parasitics information can be reused and annotated to the simulation netlists. There are two major ways circuit templates are implemented, through cell libraries and symbolic programming languages. Library-based templates saves time by eliminating the need to generate layout, the method falls apart, however, if none of the set of template masters cannot completely covers the design in question. Pre-characterized standard cells [15] tie the relation between circuit performance and parameters in an explicit way. In this approach, circuit designers select from a library of circuit building blocks. The layout completed with its associated parasitic values are provided by each block. Blocks can be of various circuit topologies, or with identical schematic constructions but differ from others by their layout design. Hence, the choice of a particular set of standard cells immediately implies the layout and parasitic values of selection.

Language-based methods are more flexible in terms of design coverage, but give up efficiency in the process. In both cases, the quality of parasitic information is high, although designers are still constrained by the flexibility, quality and diversity of topology offered by the technology-dependent, pre-characterized library of functions. Maintenance effort to keep pace with the increasingly short process technology lifespan is another issue. Moreover, inter-cellular interference and global routing parasitics such as power supply IR drop and substrate coupling are not easily accounted for.

Geometric Programming uses convex constraints and performance equations as objective functions. As equation-based methods are inheritantly approximations of the actual performance space. It is incapable of describing the non-ideal effects provide insufficient details to account for non-ideal effects and inter-dependencies among device parameters. To this date, there was no proposed geometric programming-based methods that could solve post-extraction circuit problems [39].

In the next chapter, improvement over the existing techniques is proposed by offering a template-free and semi-technology independent alternative. During circuit design optimization, estimations of physical design parasitics are performed via a preliminary placement of the circuit, and then either estimate the parasitic capacitance (without routing) or compute the *change* in interconnect parasitics (with routing) by exploiting the behavior of the extraction tool. During circuit (schematic) parameter optimization, the layout can be refined by performance-driven place and route tools, as well as a novel compaction optimization [24], at the presence of simulated performance information of candidates.

CHAPTER 4

SCHEMATIC AND PLACEMENT CO-OPTIMIZATION

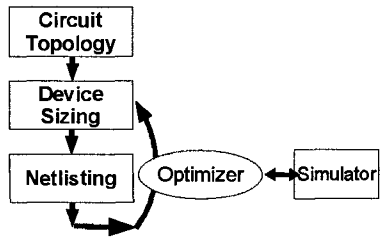
Accelerated analog circuit design cycles and the aggressive adoption of nanometer VLSI fabrication processes demand that both the efficiency and modeling techniques of conventional analog design flows must be improved. While current schematic-level optimization has sped-up the circuit design process, it does not gaurantee improvement of DSM design closure, as parasitic information is incomplete during optimization. DSM performance goals are layout-dependent, and cannot be fulfilled solely based on well-tuned schematic-level designs. In this chapter, a novel circuit and layout co-optimization design flow that improves the probability of DSM design closure is presented. The proposed design flow is tool-independent, and can be implemented by several widely available tools offered by various industrial EDA vendors.

The existing IC design flows for DSM design necessitate designers to choose between two extremes on a trade-off curve, a fast but inaccurate schematic design platform, or an accurate but slow layout design environment. Instead, this work proposes the use of parasitic estimation models to reveal DSM effects at the schematic level, such that simulation-based optimization using the schematic can provide more conclusive results. In turn, performance evaluation results are to be fed into the physical design process to aid in the decision of potential design options. A novel analog-specific compaction procedure is also incorporated in the flow to optimize the exact block placement with respect to DSM parasitic effects. Firstly, the existing independent circuit and physical design schemes are introduced, followed by the discussions of the co-optimization flow, the concept and elements that constitute it. The implementation details of co-optimization is described in details in Section 4.4. Experimental results will be given in Section 4.6.

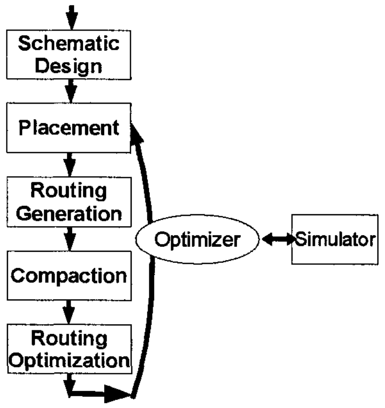
4.1 Independent Circuit and Physical Design Optimization

Due to complex design rules and conditional exceptions in the DSM era, manual circuit design relying on past experiences and rules of thumb may not achieve proper design closure. Post-extracted performance measures could potentially steer away from the target specifications even if its schematic-level circuit is satisfactory. Recently, a number of schematic-level analog circuit optimization tools [15], [64] have gained acceptance in industrial applications. They are also known as *schematic sizing tools*, because design improvements are attained by simulating the schematic netlists with transistor and other passive devices of various sizes.

The block diagram of a schematic is shown in Fig. 4–1(a). Based on the prototyping ease and accuracy of circuit simulators, they improve circuit performance through



(a) Optimization-based schematic design flow integration



(b) Trivial layout-in-the-loop automated design flow integration

Figure 4–1: Simple simulation-based optimization flows

heuristic search methods, such as the greedy, local search and in this case, simulated annealing algorithms. Search-based *empirical optimization methods* are popular because they are able to overcome the high-dimensional design space and the highly non-linear performance space typically associated with circuit design problems. The main drawback is that they do not guarantee a globally optimum solution would be found. Nonetheless, since its first introduction in 1983 [37], simulated annealing has become very popular in CAD for solving many diverse *NP Complete* problems in VLSI [28], network-on-chip task scheduling [12], etc.

Despite the success of schematic optimization, it is performed in an idealized performance space with limited DSM information, as layout effects are unknown at the schematic level. To avoid costly post-layout design rejections due to reduced performance margins and DSM effects, the basic schematic sizing tools have been extended to recognize layout-dependent parasitic effects in various ways. The foremost issue obstructing a trivial approach as in Fig. 4–1(b) is the bottleneck within the optimization loop in which each sized candidate must pass through the layout synthesis steps before parasitics can be formally extracted and simulated. Since it is impractical to create a layout for each design candidate, alternative approaches that can deduce parasitic effects without actual layout synthesis are needed.

Earlier attempts to furnish layout parasitic information for efficient circuit optimization are met by two major obstacles, namely, the restriction of the freedom of custom analog physical design and the accuracy of the extracted data. Parasitic effects can be highly *non-linear*, and the parasitic values are very much *layout-dependent*. Efforts to bypass circuit simulations with analytical models have been futile [55]. For instance, the geometric programming approach discussed in Section 3.3 approximates the *non-convex* high-dimensional circuit design space by convex constraints. This is because a convex space has a convenient property that local maxima on the convex constraint boundaries are also global maxima of the design space. The work did not account for parasitic effects and other non-idealities in the circuits. Unfortunately, accounting for them would require extensive work to determine the proper approximated convex constraints. Other ways summarized in Section 3.3 proposed the use of templates, such that extracted layout parasitics saved earlier can be annotated to the simulation netlists. However, the development of technology-dependent analog building block templates could pose maintenance problems and limit the flexibility and quality of custom analog design. Still another way adopts symbolic descriptions of the layout design. A parasitic effect extractor could then deduce the parasitic effects from the symbolic layout without actual layout generation. Among the available options, this approach could be an ideal solution, as the layout description language could be technology-independent and unrestrictive. The challenge with this technique translates to designing a general but concise description set covering all realizable custom analog layout design. A substantial paradigm shift in physical design tools is also needed. The current state of development of the layout languages has not reached this stage yet.

Since physical design revisions are expensive, multiple placement options were rarely compared in the past. Instead, analog designers rely on their experience and knowledge to inherit existing tried-and-true topologies as long as practical. Current automated design techniques such as schematic optimization [64], automated analog cell generation [46] and placement tools [9],[30],[73] are independent 'point' tools that are driven by locally available information and constraints. For example, schematic optimization is driven by performance obtained from schematic netlists simulations. Cell generation tools are concerned with packing density and aspect ratios. Floor-planning and placement tools are driven by geometric constraints, such as customized layout topologies, symmetry constraints and area consumptions, etc. As shown earlier in Fig. 1–1(a), circuit candidates are distinguished by their schematic netlist performance metrics. The best schematic candidate is then synthesized through the placement and routing steps, which focus on geometric constraints. Finally, the layout-dependent parasitics are annotated into the extracted netlist. The resultant design is sub-optimal, since layout optimization is not equipped with the best performance models and knowledge. Sensitive circuits could be steered away from the required specifications due to DSM effects.

4.2 Collaborative Circuit and Physical Design Optimization

A *co-optimization* methodology depicted in Fig. 1–1(b) that integrates the schematic optimization and automated layout placement generation tools is proposed. The integration enables mutual information exchange for parasitic-aware schematic optimization and performance-driven placement co-optimization. The proposed flow provides several options that balance execution time and model quality requirements. Fig. 4–2 illustrates the detailed implementation of the parasitic-aware design automation methodology, using a combination of off-the-shelf EDA components, such as the OPT, SIM, PLC and RTE, and estimation techniques. Designers can select from the following design optimization options based on their various needs of speed-accuracy trade-offs:

- **Schematic optimization**: (loop "`atvk`" in Fig. 4–2)

  Existing simulated annealing-based schematic optimization (OPT) using the circuit simulator (SIM) for performance evaluations.

- **Parasitic-aware schematic optimization with placement**: (loop "`ahnquvk`" in Fig. 4–2)

  Through sensitivity-based parasitic variation estimations (EST), OPT tunes device sizes on the schematic together with parasitic information estimated from placement and *virtual interconnects* (VIs).

- **Parasitic-aware schematic optimization with layout**: (loop "`ahoquvk`" in Fig. 4–2)

  Through EST, OPT tunes device sizes on the schematic together with parasitic information estimated from variation of placement and *parasitic macro-device* values.

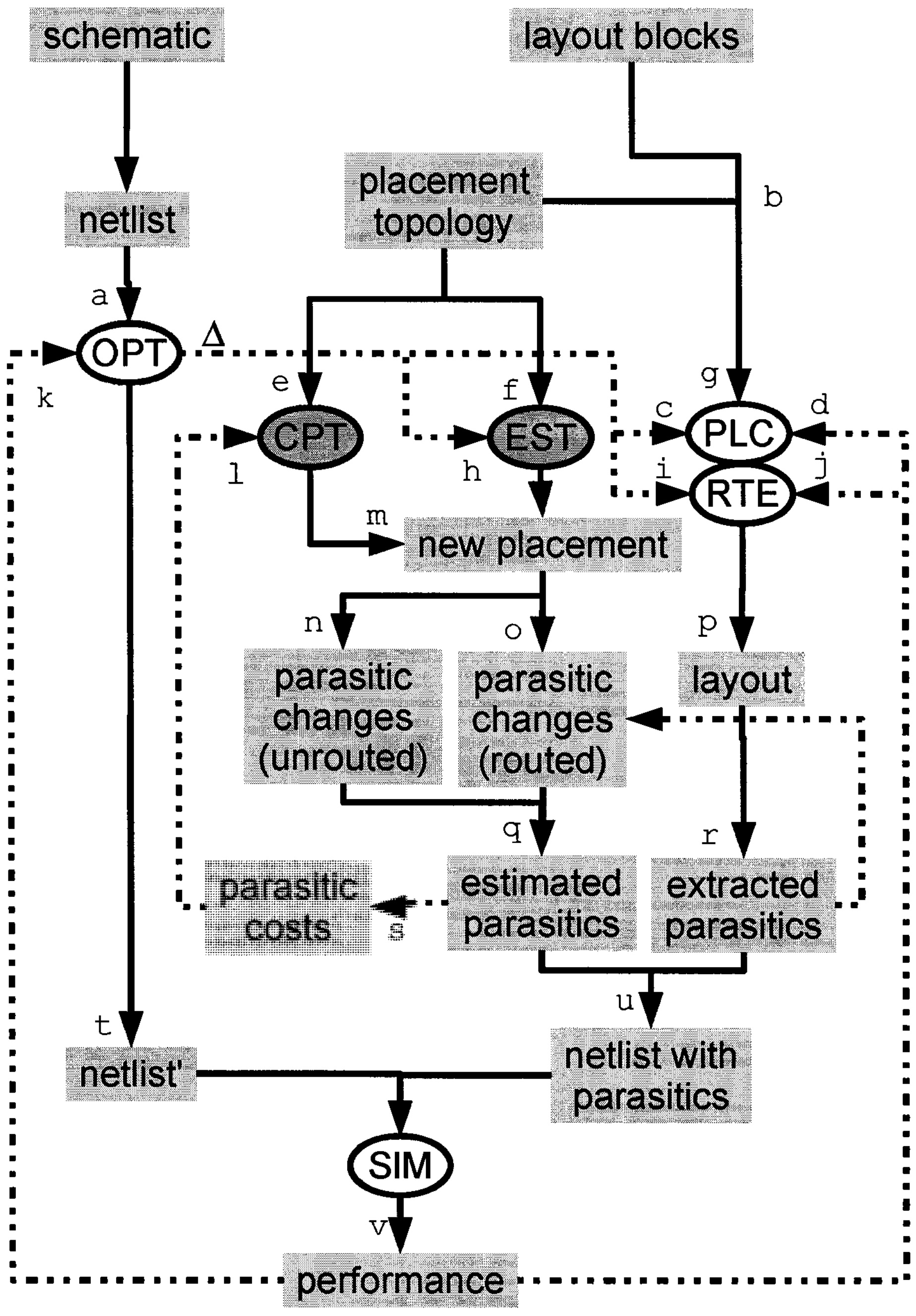


Figure 4–2: Circuit and physical design Co-optimization. Solid and dotted arrows represent design advancements and information feedbacks respectively. CPT-Compaction, EST-Parasitic Estimation, OPT-Netlist Optimizer, PLC-Placer, RTE-Router, SIM-Simulator

- **Performance-driven physical design**: (loop "`bgpruvj`","`bgpruvd`" in Fig. 4–2) Simulated performance of layout candidates are fed into place-and-route tools (PLC,RTE) to drive design decisions through paths `d` and `j`.
- **Analog compaction Optimization**: (loop "`emnqsl`" in Fig. 4–2) A novel compaction step for analog layout is proposed to optimize the balance between wire-length and substrate coupling parasitic effects. The optimization is simulated annealing-based operating on parasitic cost functions.

4.3 Efficient DSM Analog Circuit Optimization

Rather than moving from schematic design to layout design and then to post-extraction verification as in the traditional flow, this work proposes a series of co-optimization steps with progressive emphasis towards layout, as shown in Fig. 4–3. First schematic optimization is carried out. A placement is then created and the first phase of co-optimization follows. In this phase, Substrate coupling is present, but routing is absent. The interconnect parasitic estimation relies on the *virtual interconnect* (VI) model. Depending on relative block displacements, geometric variations of interconnects are translated into parasitic changes, based on parasitic sensitivity calculations. In the second co-optimization phase, the layout is routed. PMDs correlate each interconnect wire with the corresponding parasitics.

Within an *epoch* of an optimization loop, design candidates are generated through *incremental* changes of parameter values. Conservation of the netlist topology is a crucial feature, as is allows simple parametric value changes among placement candidates, avoiding parasitic re-extractions. This improves sensitivity-based estimation accuracy, and the

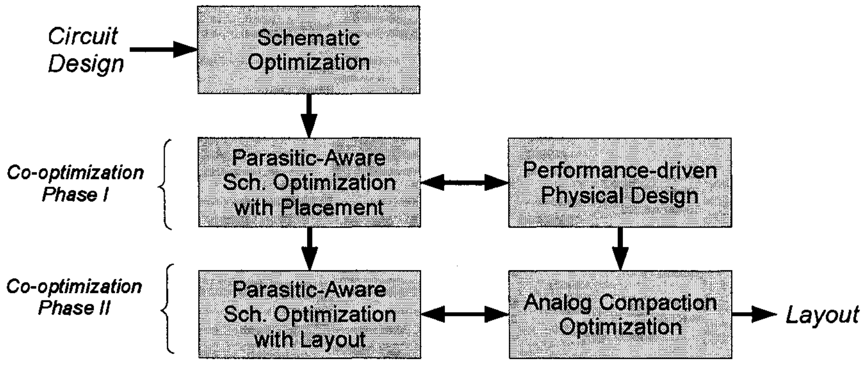

Circuit Design
Schematic Optimization
Co-optimization Phase I
Parasitic-Aware Sch. Optimization with Placement
Performance-driven Physical Design
Co-optimization Phase II
Parasitic-Aware Sch. Optimization with Layout
Analog Compaction Optimization
Layout

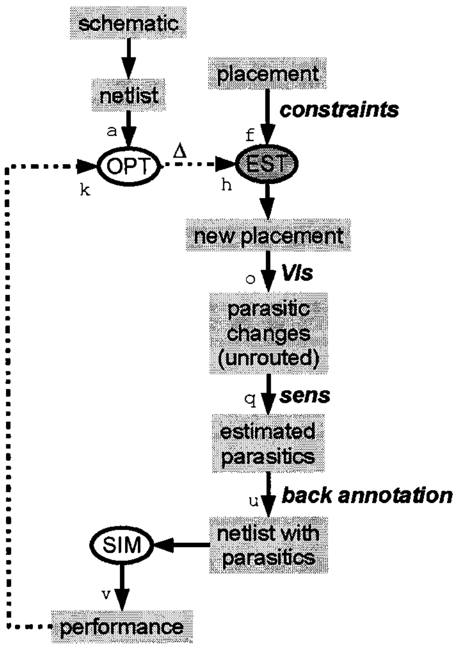
Figure 4–3: Overall circuit and layout co-optimization flow

connectivity graph of parasitic-annotated netlists can be safely shared among several candidates. Through careful control of incremental parameter changes, many costly physical routing and parasitic extraction operations can be avoided.

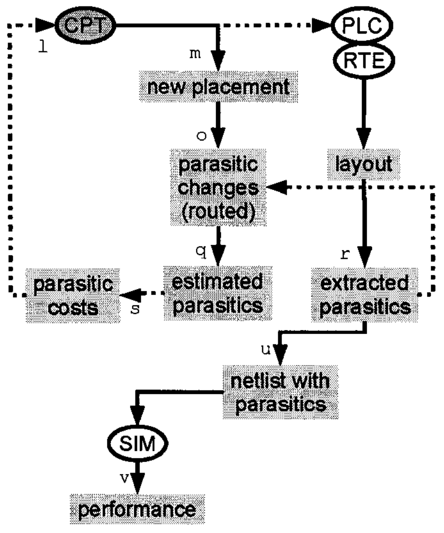
The following subsections outline the modeling techniques and components coordinated to derive the layout effect parasitic estimations. The key components involved and their adaptions to cope with designs at various stages of completion will be described in greater details in subsequent sections of the rest of the chapter.

4.3.1 Schematic Optimization

This is the conventional schematic optimization flow. The optimizer is given the design specifications, a test-bench circuit and a circuit schematic, which describes a particular circuit topology chosen by the designer. The optimizer assigns values to the design parameters of the schematic, namely the transistor dimensions and values of passive devices. Each parameter set represents a *sized netlist*, and is considered one candidate in the design space. With the help of a circuit simulator and the testbench circuit, the *performance metrics* of the prescribed circuit are obtained and compared against the required



(a) Parasitic modeling with placement



(b) Compaction optimization

Figure 4–4: Performance-Driven Physical Design - I

specifications. Based on the comparison results and previous candidate performances, the optimizer decides if the process is to be terminated, or repeated with new sets of design parameters. The optimizer can evaluate multiple candidates quickly by taking advantages of the ease of netlist modifications and the batch processing mode offered by most circuit simulators.

4.3.2 Parasitic-Aware Schematic Optimization with Placement

A placement-oriented approach is adopted to provide estimates of parasitic effects in pre-layout circuit optimization. This optimization flow is illustrated in Fig. 4–4(a). The initial schematic could be obtained from the results of schematic optimization. *Virtual* *interconnect* (VI) routing parasitics are estimated among block terminals of the given preliminary placement. The parasitics are added to the schematic netlist and then simulated. As the optimizer modifies the design parameters, the block sizes and hence the placement as well as the VIs are also updated. As will be explained in Section 2.10.2, VIs provide a deterministic basis to furnish accurate and realistic parasitic data for large number of placement topologies without performing expensive routing and extraction operations. Parasitic effects for the VI model are deduced from actual extracted parasitics and parasitic sensitivity data. Substrate coupling and interconnect parasitics are modeled in this scheme.

4.3.3 Analog Compaction Optimization

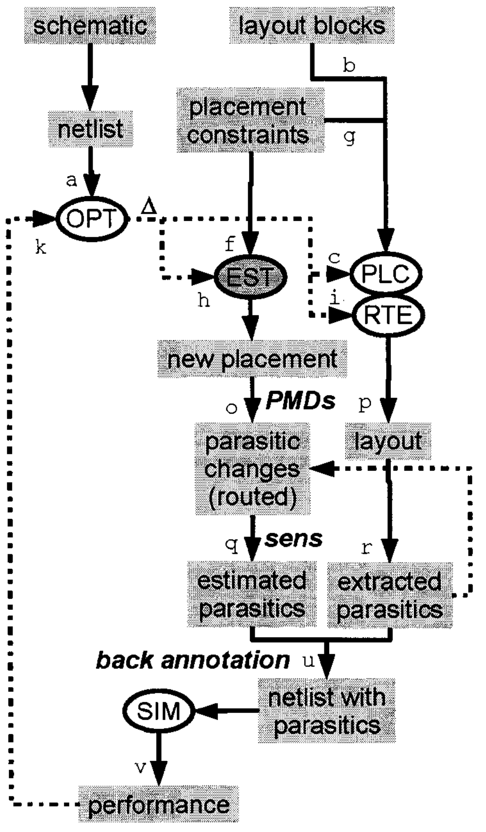
Multiple initial placement topologies of *soft* blocks are drafted manually or by automated analog cell generation [46] and placement tool [9]. After the block sizes and placement are determined, routing candidates are generated by reusing the routing model information of the corresponding placement stage. The use of parasitic sensitivity data allows comparison of performance merits among various routing configurations. Finally, the *absolute* positions of circuit blocks and guard bands are optimized in the compaction phase. Implementation details are given next, and thorough discussion of the compaction optimization process will be given in Section 4.5.

4.3.4 Parasitic-Aware Schematic Optimization with Layout

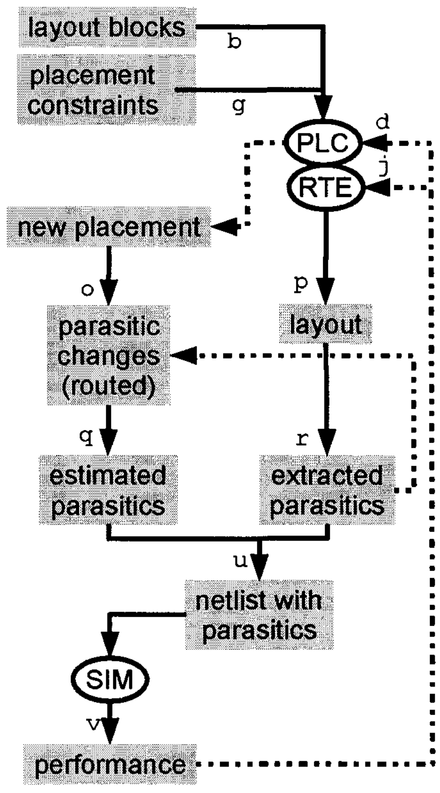
As shown in Fig. 4–5(a), candidate performance is evaluated by the circuit simulator, which accepts netlists containing the design parameters and layout parasitics. Parasitic effects are estimated through a sensitivity model based on a routed layout. To avoid excessive physical design overhead, each synthesized layout is reused a number of iterations, where parameter variations are translated into geometric variations and then mapped to parasitics with the help of sensitivity data specific to the technology in use. To enhance parasitic update efficiency. The parasitics for each interconnect object is encapsulated into *parasitic macro-devices* (PMDs). This will be further discussed in Section 2.10.1. Only the affected PMDs as a consequence of geometric variations will be updated. In performance simulation, substrate and interconnect parasitics are extracted and will be update, while cross-coupling parasitics are extracted will not. The first or second-order sensitivity model can be used. Sensitivity data is extracted from a template (Fig. 2–8) containing interconnects of various turns, lengths and widths, and is only needed once per fabrication process.

4.3.5 Performance-Driven Physical Design

A side-effect of running circuit optimization interleaved with physical design is that a greater degree of freedom in physical optimization is granted, based on performance information acrross multiple design candidates in the manner depicted in Fig. 4–5(b). Instead of optimizing on physical designs based on a circuit design, physical design could also alter circuit devices if the consequential effects are known. However, implementation of physical synthesis tools would not be covered here as they are beyond the scope of this thesis.



(a) Parasitic modeling with layout



(b) Performance driven physical design

Figure 4–5: Parasitic-aware Design Automation - II

4.4 Parasitic-Aware Performance Evaluations

During optimization, decisions guiding the changes of device parameters are determined either by performance *estimation* or *simulation*. Table 4–1 outlines the types of parasitic effects considered by the various design flows mentioned in Section 4.3. The changes are propagated to the placement level, and then to the routing parasitics, which are governed by the PMD models. In performance estimation, geometric changes are mapped to

Table 4–1: Types of Parasitic Effects Modeled

FLOW TYPE	SUBSTRATE COUPLING	INTERCONN. RC	COUPLING CAP.
Schematic optimization	No	No	No
Parasitic-aware schematic optimization with placement	Yes	Estimated	No
Analog compaction optimization	Yes	Estimated	No
Parasitic-aware schematic optimization with layout	Yes	Extracted / Estimated	Extracted
Performance-driven physical design	Yes	Yes	Yes

parasitic changes and then to performance changes via parasitic and performance sensitivity values. In performance simulation, geometric changes are mapped to parasitic changes via parasitic sensitivity values. The affected parasitics in the post-extraction netlist are then updated, and the netlist is simulated. In both cases, only the affected interconnect parasitic models (line 7) are updated through the interconnect estimation algorithm from Fig. 4–6. The choice between performance estimation and simulation is hinged on the relative importance of accuracy and simulation speed. The following subsections outline the sensitivity-based computation procedures.

4.4.1 Sensitivity-Based Performance Variation Estimation

During optimization, design parameters are adjusted continuously in hope for performance improvement. The use of sensitivity information is crucial to optimization [7], [47], as it is simple to compute and guide the desirable direction the parameter values

```
while(1)
 if blocks have been resized
  send blocks to synthesis tool
  obtain placement from layout synthesis
 end if
 from placement determine terminals MHS MVS
 IntconnEstimate estimates parasitics
 annotate parasitics to circuit netlist
 simulation-based schematic optimization
 if new schematic cell sizings exceed blocks
  resize blocks
 end if
end while
```

Figure 4–6: Algorithm: Parasitic-aware schematic optimization

should change. The Taylor series expansion of the performance function $K(\mathbf{d})$ in terms of design parameters $d_i \in \mathbf{d}$ is the performance sensitivity function for d_i. Using only the 1st derivative of K, *gradient* is the most common implementation of the sensitivity function. It is essentially the linear extrapolation of the performance function. This subsection presents procedures to compute the performance sensitivity with respect to layout parasitics introduced by design parameters. Let K_{sch} denotes a schematic-level performance measurement. It is treated as the cost function during optimization. Performance K_{sch} can be expressed in the form

$$K_{sch} = f_{sch}(\mathbf{d}) \tag{4.1}$$

where f_{sch} is any differentiable function and $\mathbf{d} = d_1, ..., d_k$ are design parameters. For multiple objectives optimizations, (4.1) is defined for each performance measurement. In order to simplify the ongoing discussion, single performance objective optimization is assumed. In practice, (4.1) is solved by circuit simulators except for the simplest cases. During sensitivity-based optimization, the algorithm continually evaluates current performance $f_{sch}(\mathrm{d}^{(n)})$ and determines if it meets design criteria. If it does not, a new search direction based on the sensitivities of performance K_o with respect to the set of $d_i \in \mathrm{d}^{(n)}$, or $\frac{\partial K_o}{\partial d_i}$, are determined to guide the optimizer to the next simulation point $\mathrm{d}^{(n+1)}$ and the process repeats. Now, suppose the circuit has m parasitic macro-devices H and k parameters d belong to some schematic devices $\mathbf{D}$, the post-extraction circuit performance becomes $K = f(\mathbf{H}, \mathrm{d})$, where $\mathbf{H} = H_1, ..., H_m$ and $\mathrm{d} = d_1, ..., d_k$. Some $h_i \in H_p$ are influenced by device parameters $d_j \in D_q$ due to layout if H_p and schematic device D_q are adjacent, as in Fig. 2–20. Otherwise, they are independent. Additionally, the performance sensitivity due to parasitics induced by d is also found. In general,

$$\frac{\partial K}{\partial d_j} = \sum_{i=1}^{m} \frac{\partial f}{\partial H_i}\frac{\partial H_i}{\partial d_j} + \sum_{i=1}^{n} \frac{\partial f}{\partial d_i}\frac{\partial d_i}{\partial d_j} \tag{4.2}$$

$$= \sum_{i=1}^{m} (\frac{\partial f}{\partial H_i} \sum_{h_l \in H_i} (\frac{\partial H_i}{\partial h_l}\frac{\partial h_l}{\partial d_j})) + \frac{\partial f}{\partial d_j} \qquad \forall d_j \in \mathrm{d}\,. \tag{4.3}$$

Noting that $\frac{\partial d_i}{\partial d_j} = 0$ for $i \neq j$ and $\frac{\partial f}{\partial d}$ treats all H in f as constants. Rearranging the summation terms and note that $\frac{\partial H_i}{\partial h_j} = 0$ if $h_j \notin H_i$, rewrite (4.3) as

$$S^f_{d_j} = \tilde{s}^f_{d_j} + s^f_{d_j}\,, \qquad \forall d_j \in \mathrm{d} \tag{4.4}$$

where $S^f_{d_j} = \frac{\partial K}{\partial d_j}$, $s^f_{d_j} = \frac{\partial f}{\partial d_j}$ and $\tilde{s}^f_{d_j} = \sum_{l=\langle p_j \rangle}(\frac{\partial h_l}{\partial d_j}(\frac{\partial f}{\partial H_i}\frac{\partial H_i}{\partial h_l}))$. This summation corresponds to a subset of PMDs, indexed by $\langle p_j \rangle$, that are *immediate* neighbors of the schematic device D that d_j belongs. Here, $\tilde{s}^f_{d_j}$ represents the performance sensitivity contributed by parasitic effects introduced by d_j. The inner summation in (4.3) is simplified, since there is a *unique* PMD H_i that encapsulate any given h_j. Evaluating (4.4) enables the optimizer to predict the performance impact of parasitic effects introduced by a given set of parameters, as well as giving grounds when parasitic estimation is justified. The next subsections describe how $\frac{\partial f}{\partial H}$, $\frac{\partial H}{\partial h}$ and $\frac{\partial h}{\partial d}$ are computed. For convenience, the subscripts are dropped in the following discussion when no ambiguity arises.

4.4.2 Performance Sensitivity Computation

The following explains how $\frac{\partial f}{\partial H}$ and $\frac{\partial H}{\partial h}$ are computed. The number of parasitic devices h in an extracted circuit is of $O(n^2)$, where n is the number of schematic devices. Computing the performance sensitivity directly from the definition

$$\frac{\partial f}{\partial H} = \lim_{\Delta h \to \infty} \frac{f(h + \Delta h) - f(h)}{\Delta h} \tag{4.5}$$

is computationally expensive, as it requires a large number of layout extractions and simulations. Thus, parasitic macro-devices (PMDs) H that group interconnected parasitic devices into clusters are defined, so that $\frac{\partial f}{\partial h} = \frac{\partial f}{\partial H} \cdot \frac{\partial H}{\partial h}$. As the number of PMD H is much less than the number of individual parasitic device h, first obtain $\frac{\partial f}{\partial H}$ by simulation, then apply (4.5). $\frac{\partial H}{\partial h}$ is the sensitivity of H with respect to one of its constituent h. For $\frac{\partial H}{\partial h}$, the computationally efficient *adjoint* method is used. Note that H is a relatively small passive network, and forming its adjoint system is straight forward. Once the solutions of the system H and its adjoint system H^a are found, the sensitivity for each h can be expressed in terms of these solutions without any additional computations [68]. Let the system of equations of H be

$$\mathbf{H}\mathbf{X} = \mathbf{Y} \tag{4.6}$$

where $\mathbf{X}$ are nodal voltages and branch currents and $\mathbf{Y}$ are sources. Both of them are known from the simulation results in computing $\frac{\partial f}{\partial H}$. Let the circuit measurement at an output ports $\phi_1, ..., \phi_n$ of H be some generalized function

$$\phi = g(\mathbf{X}, \mathbf{h}) \tag{4.7}$$

for each ϕ_i. Define the adjoint system to be [68]

$$\mathbf{H}^t\mathbf{X}^a = -(\frac{\partial \phi}{\partial \mathbf{X}})^t \tag{4.8}$$

then once (4.8) is solved, the sensitivies of ϕ with respect to all parasitic devices can be expressed in terms of $\mathbf{X}$ and $\mathbf{X}^a$. Suppose the nodal voltages of the ports of parasitic device h in H are V_1, V_2. Its sensitivity for ϕ is then

$$\frac{\partial \phi}{\partial h} = s(V_1^a - V_2^a)(V_1 - V_2) \tag{4.9}$$

where s is the operating frequency if h is a capacitor or inductor, or s is zero if h is a resistor [68]. To summarize, the computation procedures for $\frac{\partial f}{\partial h}$ is as follows:

1. Parasitic devices in the extracted circuit are lumped as H.
2. The extracted circuit is simulated and perturbed, and $\frac{\partial f}{\partial H}$ is found.
3. Solve the adjoint system (4.8) for each H, and $\frac{\partial H}{\partial h}$ is found by (4.9).

4.4.3 Parasitic Sensitivity Computation

The sensitivity of a parasitic device towards the given design parameter $\frac{\partial h}{\partial d}$ is estimated based on models built using data collected from past layout extraction results. A dynamic parasitic model based on a multi-dimensional linear interpolation scheme [59] is employed to map design parameters d to H . The algorithm first checks if the request design parameters d is in the database. The extracted parasitic macro-device is returned if it is found. Otherwise, two adjacent data points from each dimension (parameter) that bounds the request data point are used to interpolate a linear approximation that equates the gradient at the required point along that dimension. If the request value is outside the range between the smallest and largest stored points in any dimension, no solution will be returned. The algorithm then sends the circuit for layout synthesis and extraction. Later, the extracted data are entered into the database. A database maintenance routine eliminates non-essential data points and keeps the size of the database manageable. The overall algorithm is shown in Fig. 4–7.

4.5 Performance-Driven Analog Compaction Optimization

Analog circuit compaction is fundamentally different from the classical compaction problem for digital circuits. While a compact placement that minimizes its total silicon area is advantageous for digital circuits, analog circuits may benefit from additional separation spaces among circuit blocks for interference reduction. Maximizing the block separation distances may improve insulation from inter-block substrate coupling, but parasitic

```
ParasiticEffect( design_pars ):
 foreach (i, design_par) in design_pars:
  dev = GetDev( design_par )
  parasitics = GetAdjacentParasitics( dev )
  foreach parasitic in parasitics:
   PMD = GetPMD( parasitic )
   sens = GetSensitivityFromModel( dev, parasitic )
   sens *= GetPMDSensitivity( parasitic )
   sens *= GetPerformanceSensitivity( PMD )
   sens_sum += sens
  end foreach
  layout_perf_sens[i] = sens_sum
 end foreach
 return layout_perf_sens
```

Figure 4–7: Algorithm: Parasitic sensitivity calculation algorithm

resistance and coupling capacitances of critical nets may also increase. Therefore, compaction optimization for analog circuit needs to find the balance among block separation distances, area and interconnect lengths.

The compaction optimizer accepts a preliminary physical analog design placement [66],[40],[8] with symmetry properties specified. Each block is allowed to shift within specific constraint boundaries, to determine the optimum trade-offs between coupling reduction, interconnect wire length and area consumption. Placement topology and symmetry constraints are enforced throughout the course of optimization. Each block can possess self- or mirror- symmetry properties with another block and the corresponding symmetry axis in its property list.

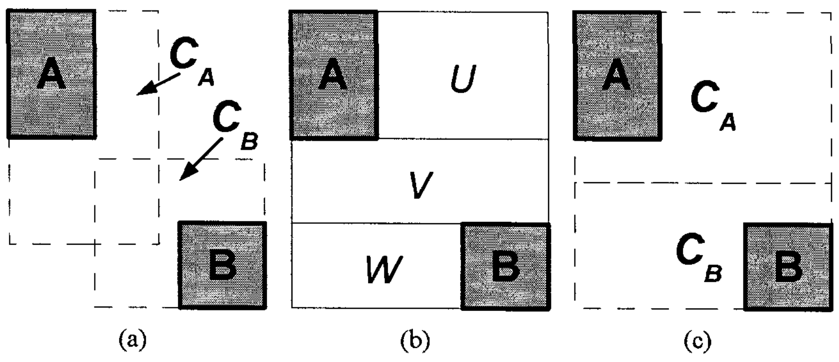


Figure 4–8: (a) Overlapping block placement constraints (b) MHS partition (c) Non-overlapping constraints derived from MHS partition

4.5.1 Non-overlapping Compaction Constraints

In optimization, input parameters are varied within a specified range, forming a unique candidate. Its quality is then ranked against other candidates, among which the best ones are obtained. In compaction, the input parameters are the geometric locations of the device blocks. Each block is allowed to shift within its rectangular placement constraint boundary. If block placement constraints are allowed to overlap, block positions will have dependency on each others. In Fig. 4–8(a), the placement constraints C_A and C_B overlap each others, but only one block is allowed to occupy any given location.

In simulation-based optimization, candidate throughput is crucial to the optimization quality. In order to avoid blocks overlap, either a a positional overlap violations checking step, or a conditional block displacement scheme are needed. However, these would place overhead during optimization that undermine its efficiency significantly. A more restrictive scheme with non-overlapping placement constraints is employed here to allow effective combinatorial optimization. Fig. 4–8(b) shows the MHS partition of blocks A and B creates 3 vacant tiles U, V and W. A non-overlapped placement constraint scheme can be constructed by horizontally bisecting V. The two halves are then merged with U and W to form the placement constraints C_A and C_B respectively, as shown in Fig. 4–8(c). To preserve symmetry properties throughout compaction, the placement constraints of mirrored block pairs are also mirrored. *Definition 1* defines the placement constraint specifications derived from the MHS partition. The derivation is efficient, as constraint boundaries of each block are independent of others. It also accounts for self, mirror and perfect symmetry properties, such that they are enforced throughout every possible compaction.

***Definition**: Given the MHS partition MHS(P) of a layout placement P, the placement constraint C_p for each circuit block p in P is defined by a rectangular boundary computed as follows:*

1. If p has no symmetry constraints, then
    - The vertical boundaries on each side of C_p are the vertical bisectors of the narrowest neighboring vacant tiles. If no blocks are on their opposite sides, then C_p spans the entire widths of the vacant tiles.
    - Similarly, the horizontal boundaries of C_p extend to the horizontal bisectors of the of the shortest neighboring top and bottom vacant tiles. If the opposite sides of them are not device blocks, then C_p extend to the entire heights of the shortest neighboring vacant tiles.
2. If p is perfectly symmetric, C_p is equivalent to the boundary of p. That is, the placement of p is fixed and would not be optimized.

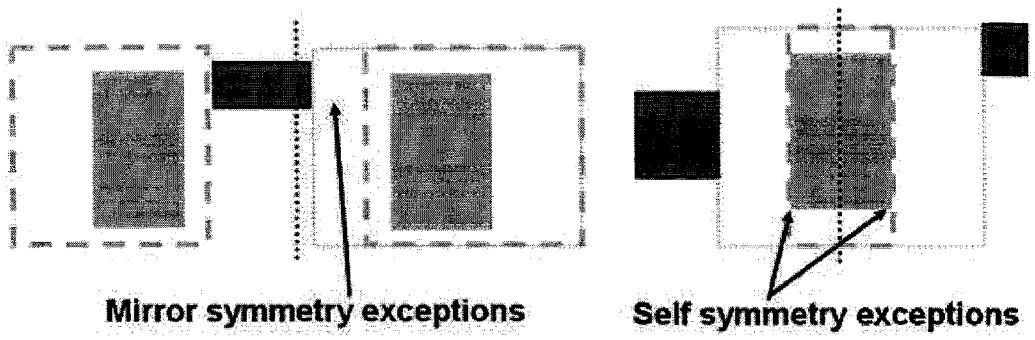


Figure 4–9: Mirror- and self-symmetry exceptions on compaction constraints

3. If p is self-symmetric about vertical axis l, then C_p is derived according to rule 1 stated above, except that its vertical boundaries are the same as the vertical boundaries of block p.
4. If p has mirror symmetry with block p' about vertical axis l, then $C_p = C_p \wedge mirror(C'_p, l)$, where C_p and C'_p on the right side of the equation are derived according to rule 1 stated above.

Fig. 4–9 shows placement constraints derived for various blocks with symmetry constraints. The left block pair has mirror symmetry about the central vertical axis, its constraint boundaries are the intersection of itself and the mirror image of its partner. In other words, the rectangular placement constraints of a mirrored-pair are the mirror image pair of the more restrictive rectangle. Hence, the left constraint boundary of the right side block only extends to the same extent towards the symmetry axis as its left partner. The other block is self symmetric and thus only vertical displacements are allowed.

4.5.2 Simulated Annealing

Simulated annealing is a widely used optimization approach to overcome local minima in highly nonlinear design spaces. Implementation details of the analog compaction scheme are summarized by the algorithm in Fig. 4–10. The algorithm begins at an arbitrarily chosen candidate in the compaction space (line 4). A new candidate is drawn from a multi-dimensional bounded Gaussian distribution function $N(\mathbf{x}_0, \sigma)$, where the mean $\mathbf{x}_0$ is the current parameter values, and σ is controlled by the *neighborhood radius* (line 7). At radius value of 1.0, the entire range spans 6σ. Lowering the radius results in a more focused search. For each newly generated candidate, its impact to the circuit performance is evaluated and compared to the lowest cost candidate currently found (line 9). The candidate is accepted if the cost difference ΔC satisfies

$$e^{-\Delta C/T} > P \tag{4.10}$$

where P is drawn from the uniform distributed interval $[0, 1]$. If the new cost is lower, $e^{-\Delta C/T} > 1$, acceptance is thus guaranteed. A linear cooling schedule is used, which gradually lowers the virtual temperature T as the search progresses (line 16). This reduces the likelihood of (4.10) to be satisfied for steps that increases cost $\Delta C > 0$. Upon reaching the pre-determined number of iterations, the algorithm returns with a list of compaction configurations incurring the lowest costs.

4.5.3 Performance Cost Functions

Circuit simulators facilitate calculation (e.g. `.SENS` in SPICE) of both the DC operating-point and AC small-signal sensitivities $\partial f / \partial \mathbf{x}$ of an output variable f with respect to any circuit parasitic values $\mathbf{x}$. The cost function with respect to f is

$$C_f(\mathbf{x}) = \frac{1}{f + \partial f / \partial \mathbf{x}^t \cdot \Delta x} \tag{4.11}$$

```
procedure compaction update:
//Input:  MHS, symm.  comp.  constraints
//Output:  list of best compactions
(temperature,intconn,cand) = Init(MHS,constraints)
for i from 1 to number of iterations:
 newname = Namer()
 newcand = Nextcand(bounds,cand,radius)
 newcost = Cost(Placement, interconn, cand)
 if AcceptHillClimb(temperature,cost,newcost):
   bestcands.append((newcost,newname,newcand))
   bestcands.sort()
   cand = newcand
   if length(bestcands) > numtopcand:
    bestcands = bestcands[1:numtopcand]
    cost = LowestCost(bestcands)
   temperature = Cooling(temperature)
 return bestcands
```

Figure 4–10: Algorithm: Analog layout compaction optimization

Δx results from block displacements are computed for the substrate coupling network and interconnect parasitics respectively. To improve efficiency, the interconnect routing would not be updated between compaction iterations. As optimization progresses, relative positions among circuit blocks changes the parasitic model values.

4.6 Implementation and Experimental Results

The proposed optimization flow is depicted in Fig. 4–11. It consists of components introduced earlier and is compatible with the following industrial standard CAD components:

- Cadence Virtuoso Schematic and Layout Editor
- Cadence Analog Artist Simulator Interface for Spectre and HSPICE
- Cadence Assura Parasitic Extractor
- Synopsys Star RCXT Parasitic Extractor

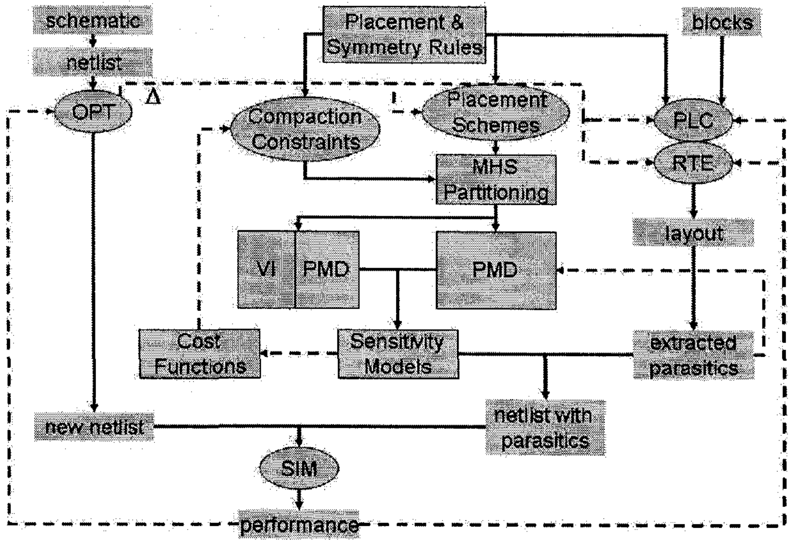


Figure 4–11: Proposed optimization flow block diagram with standard CAD components: OPT-Netlist Optimizer, PLC-Placer, RTE-Router, SIM-Simulator

The main body of software component of this work is implemented in Python [53], a popular open-source scripting language. A portion of it is written in SKILL, the proprietary script language of Cadence Design Systems. The optimization procedures based on simulated annealing are written in Matlab. The proposed flow keeps the existing Cadence designer's interface intact and employs the circuit simulator for performance evaluations. The circuit simulators used in the development of this work are:

- SPICE Circuit Simulator
- Cadence Spectre Circuit Simulator

In certain occasions such as PLL design, accurate simulation of jitter response is still tedious, due to the need to observe long durations involving large numbers of small simulation time steps for capturing noise impulses and high signal carrier frequencies.

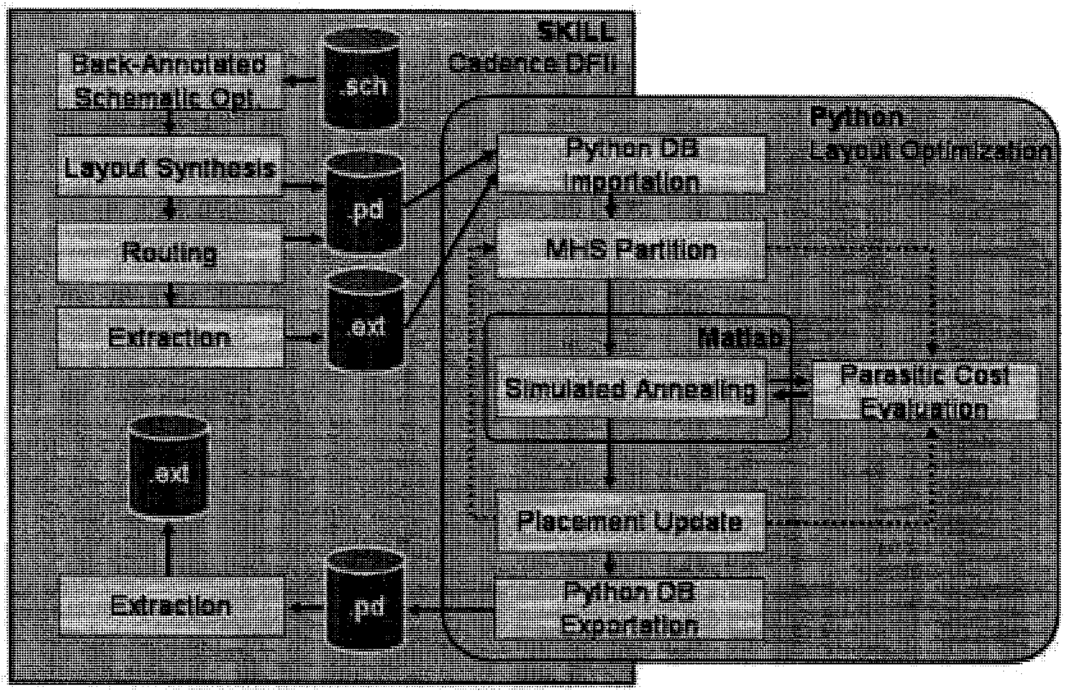


Figure 4–12: Proposed optimization flow implemented in Python, SKILL and Matlab

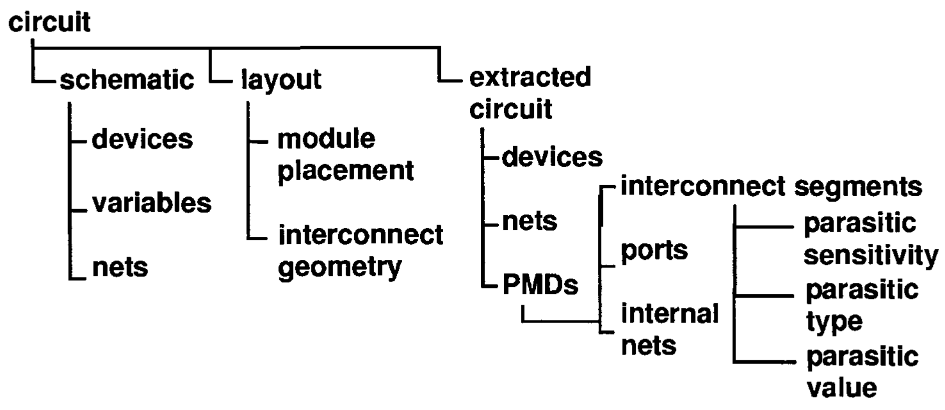


Figure 4–13: Circuit data structure implemented in Python

Each circuit and physical design is first established in Cadence design database object. An interface that reads and writes Cadence design flow object data enables the transfer of schematic, layout and extracted designs between the Cadence and Python environments. Fig. 4–12 illustrates the interactions among the components. It stores circuit, placement and extracted netlist as graph objects, and facilitates utilities to add, modify or remove circuit components. The design is manipulated in Python as a graph-based object. The underlying software database structure for each circuit is illustrated in Fig. 4–13. .

4.6.1 Accuracy of Parasitic Estimation

To validate the accuracy of the parasitic estimation algorithm, 100 layout-sensitive DSM circuits are created and extracted using the Synopsys Cadabra cell creation and Star-RCXT extraction tools. Extracted resistances and capacitances and their sensitivities for `metal 1` wires of various widths and lengths for CMOS 0.18μm are shown in Fig. 4–14(a) and Fig. 4–14(b). A test chip containing switching noise generators and probing pads at several set distances apart, connected to various diffusion areas, n-well and the

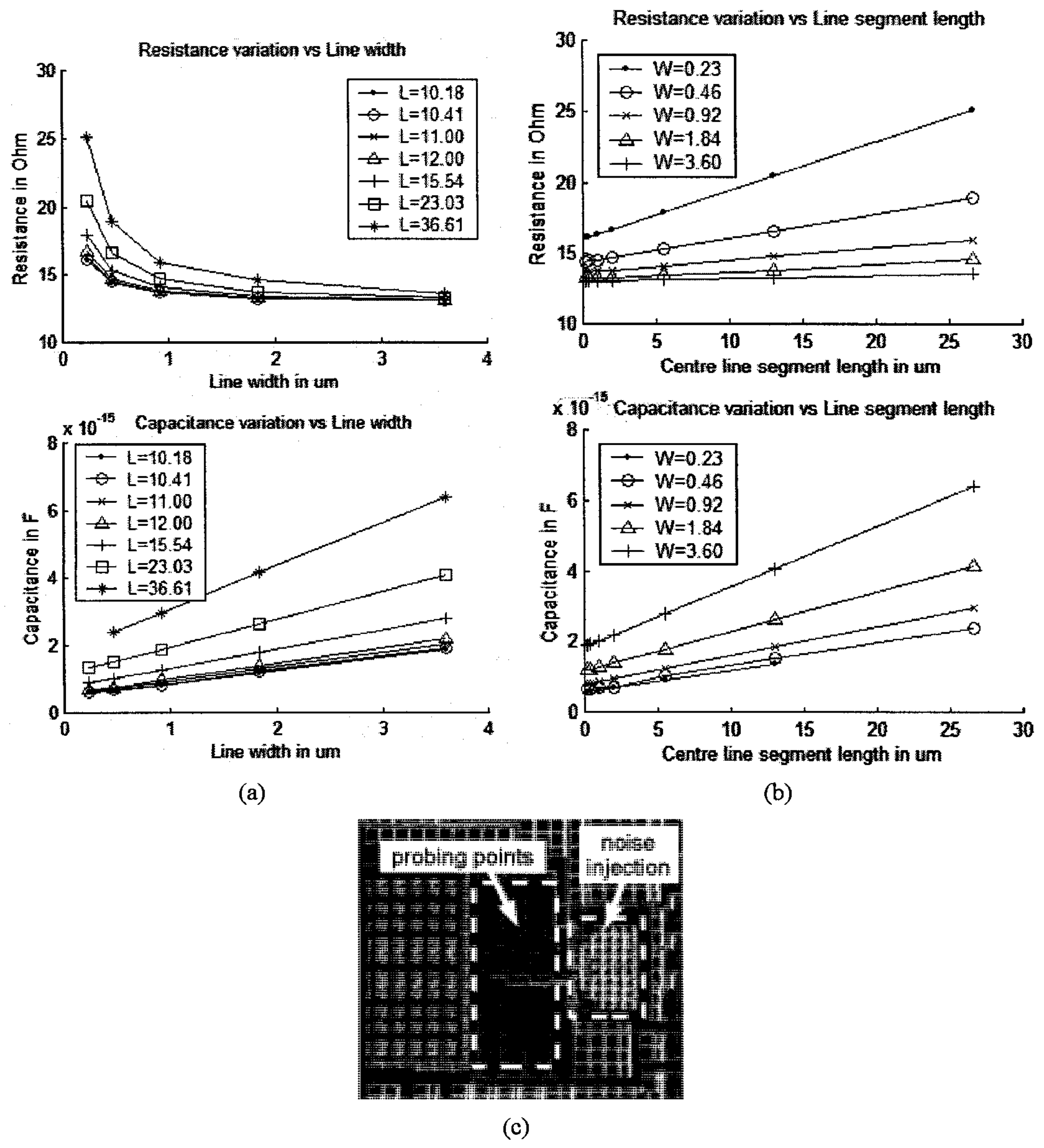
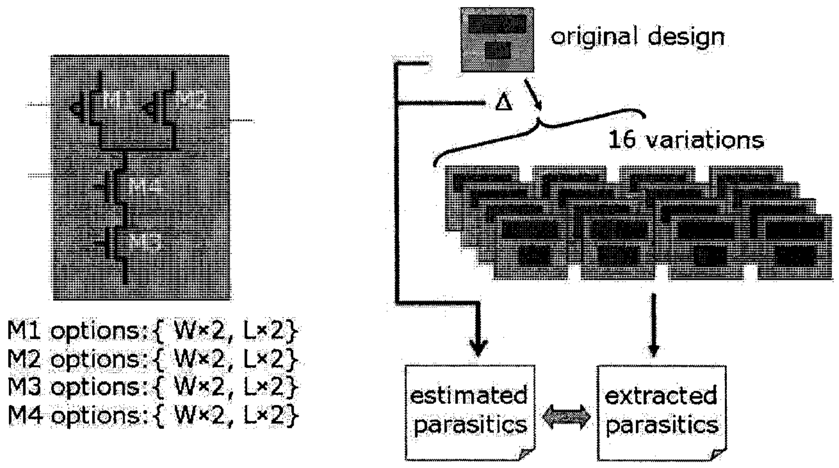
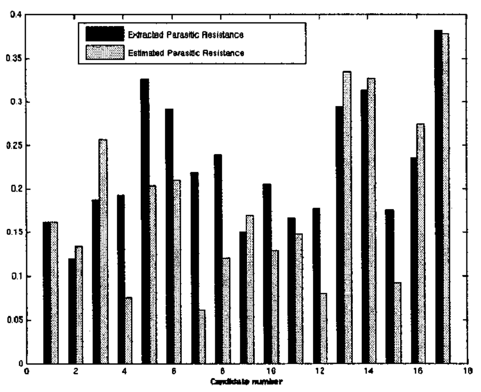


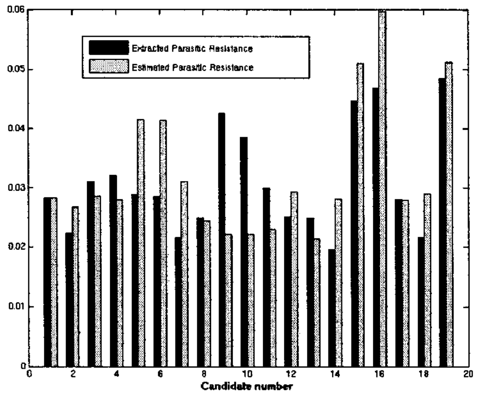
Figure 4–14: Extracted inline resistance and ground capacitance parameters (Z_{inl}) for 0.18μm for various interconnect (a) lengths, (b) widths, and (c) test chip for substrate parameters measurement experiments



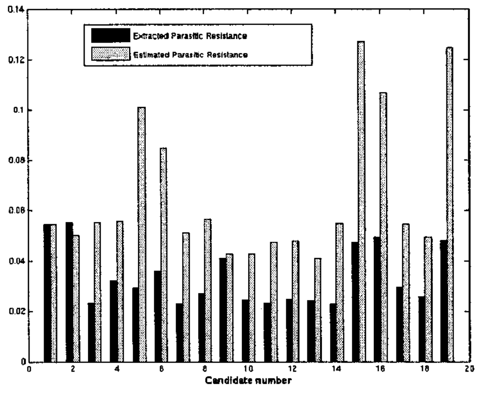
(a) Experimental Setup



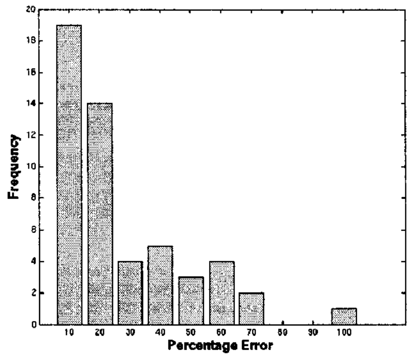
(b) Input node



(c) An internal node



(d) Output node



(e) Average estimation error histogram

Figure 4–15: Comparisons of extracted (dark bars) and estimated (light bars) Y_{11} measured at various nodes

substrate is fabricated to obtain more accurate substrate parameters, shown in Fig. 4–14(c). The estimated parasitic values using the sensitivity data are then compared with the extracted parasitics at various places. Fig. 4–15 compares the estimated and extracted input admittance (Y_{11}) at several probed locations. Statistics of the average parasitic resistance estimation error of 52 design is shown in Fig. 4–15(e). Estimated parasitic values update versus the extracted values of resynthesized layout are compared. It shows that majority of the estimation error is less than 20%, which is acceptable for is purpose as a first-order analysis technique. The execution time of the estimation algorithm is less than 5 seconds per design update, while the layout generation and extraction time consistently exceeds 100 seconds. The physical design and extraction costs are expected to increase dramatically for larger designs.

4.6.2 A Circuit Design Optimization Example

This subsection demonstrates the optimization of an analog amplifier. The process begins with a manually selected differential amplifier circuit topology. A schematic with defined interconnection of transistor and passive devices are entered into the schematic editor. The tentative device sizing parameters are then optimized. The schematic optimization is carried out by an annealing-based optimizer in association with the circuit simulator. Based upon the transient, AC and DC simulation results, multiple evaluation functions are defined to measure the top-performing candidates. The candidates are compared against the specifications and against each others across these objectives. Fig. 4–16 illustrates the graphical view for convenient comparison of design candidates across multiple objectives within the Synopsys Circuit Explorer tool. Each candidate is represented

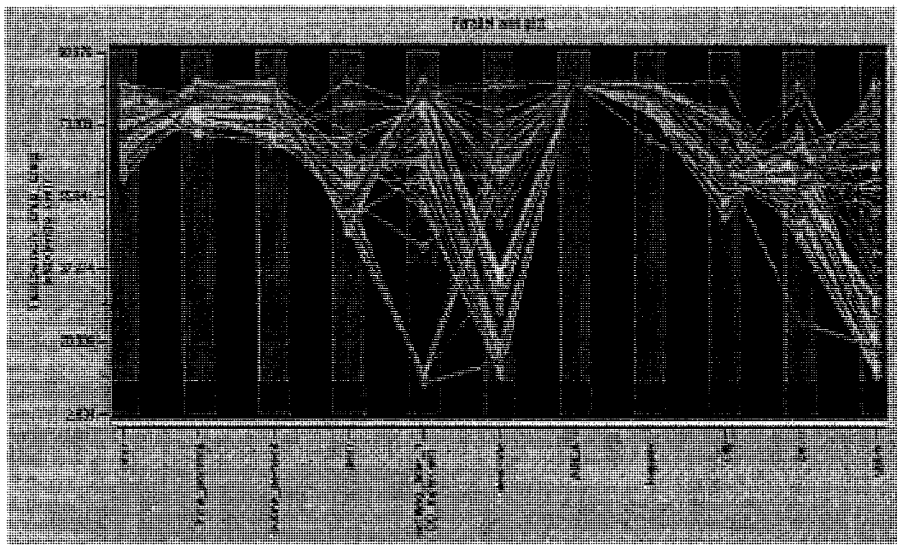


Figure 4–16: Comparison of design candidates across multiple objectives in Synopsys Circuit Explorer

by one line connecting its various performance values, where their performance trade-offs can be visually inspected.

Next, the relative block placement is manually defined, or generated automatically via automated placement procedures. Fig. 4–17 shows the query window in the Cadence layout editor, where block symmetry properties are defined and saved into the circuit design database. The routing and substrate parasitics are derived based on the relative block positions and fabrication process data, using the sensitivity-based estimation techniques outlined in Section 2.9.2 and 2.10. For instance, Fig. 4–18 shows different approaches to obtain the results of 3 design parameter changes of another design, based on the original $W/L = 7.5\mu/.75\mu$ values. The simulation results and the time required to achieve them

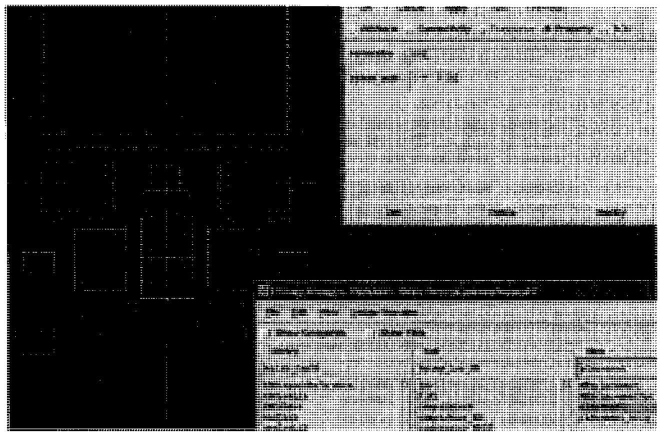


Figure 4–17: Query window for block symmetry properties definition

W/L of pMOS block	7.5u/.75u	7.5u/.5u	5u/.5u	5u/.75u	Total time (excl. design edit)
Schematic	3.618	3.118	2.356	2.707	15.56 (simulation)
Extracted Parasitics	3.896	3.259	2.567	3.024	82.56 (extraction + simulation)
Estimated Parasitics	3.896	3.323	2.819	3.152	32.94 (estimation + simulation)

Figure 4–18: Comparison of simulated A/D Conversion delay (ns) of different design variables (pMOS W/L) and their average execution time (s) via different circuit models

with (i) no layout parasitic extraction, (ii) via parasitic extraction, and (iii) via sensitivity-based parasitic estimation method without explicit layout editions are compared. Additional time for layout editions is required in case (ii) to be used by the extractor, but not needed for case (iii). The extracted and estimated parasitics with the new connectivity data are back-annotated to the schematic as a new schematic design. This completes the layout parasitic effects modeling process. With the additional parasitics information derived from the physical design, the circuit can be re-simulated to obtain the post-extracted performance data. The schematic can again be further optimized in the post-extracted performance space. The completed amplifier layout is shown in Fig. 4–19.

Fig. 4–20 compares the extracted simulation of the layout shown in Fig. 4–19 against the performance metrics of the optimum solutions after optimization, whether parasitic modeling has been used. It shows that the optimum solution with parasitic knowledge adheres more towards the performance of the extracted solution than that without. While both optimum candidates meet the required specifications in this example, the conclusion may differ for other instances if applied to a more parasitic-sensitive circuit under tighter specification requirements. This example shows that the parasitic-aware optimization scheme has merit for future layout-sensitive DSM designs.

The concept of co-optimization enables the circuit and physical design optimization procedures to be interlaced. In contrast, traditional design automation flows are sequential. Layout optimization is permitted to begin after circuit optimization has fully completed. The number of epochs carried out before switching between the processes is also arbitrary. This ratio represents the balance between the efficiency and accuracy parasitic model parasitic-aware schematic optimization. If speed is of high priority, several epochs of the schematic optimization can be performed per visit of the physical effect modeling loop to amortize its cost. This is due to the relative ease of schematic optimization. If, however, modeling accuracy is of higher priority, multiple instances with differing incremental geometric changes can be modeled by means of parasitic sensitivity. For each schematic design, multiple back-annotated instances of physical designs are netlisted, simulated and compared against each other. The efficiency of performance-driven layout synthesis can be further improved with commercial place-and-route tools.

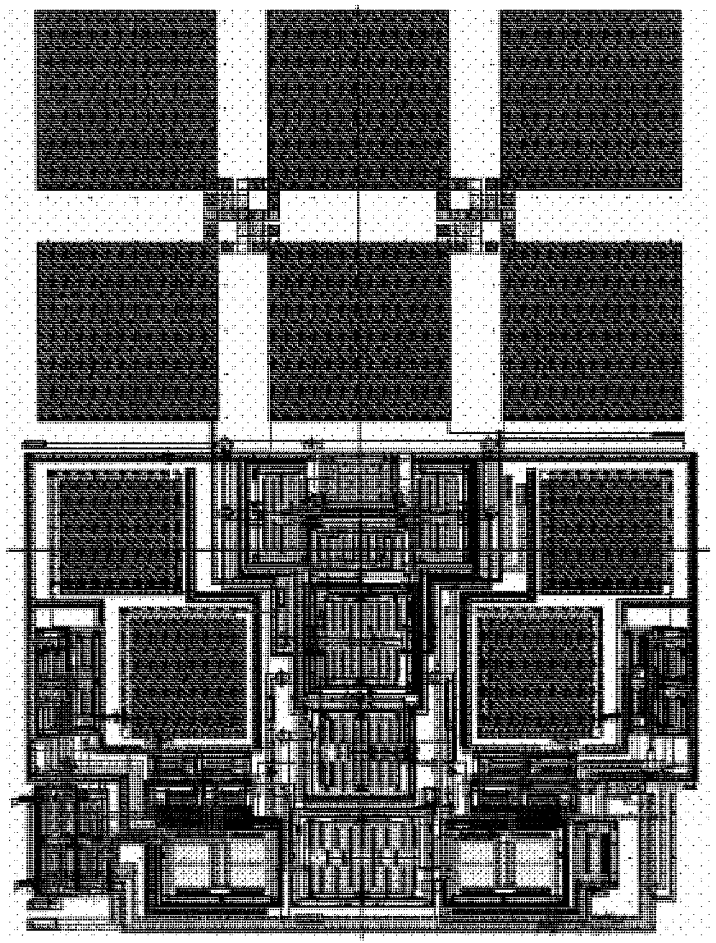


Figure 4–19: Completed differential operation amplifier layout

Quantity	Spec.	Opt. cand. without para. modeling	Opt. cand. with para. modeling	Post-extraction simulation
Bandwidth -5dB	>5 M	10.43 M	10.15 M	10.29 M
Bandwidth -3dB	> 10 M	17.07 M	16.87 M	16.96 M
Peak AC	< 0.3	508.7 μ	138.6 μ	116.1 μ
Noise at 10kHz	< 70 n	28.51 n	28.51n	-
CMRR	> 60	64.207	64.207	-
PSRR	> 45	52.325	52.353	-
Iout Sink	> 1 μ	2.514 μ	2.514 μ	2.309 μ
Iout Source	> 0.5 m	4.128 m	4.128 m	4.151 m
In SW %	> 70	84.226	84.226	-
Out SW %	> 70	38.622	38.622	-
Overshoot	< 6	865.1 m	960.1 m	-
Settling Time	< 2 μ	392.3 n	395.5 n	317.2 n
Trans. Slew Rate	> 2 M	5.77 M	5.704 M	5.654 M
THD %	< 0.25	1.371	1.373	-

Figure 4–20: Comparison of simulation and optimization performance results of a 0.18μm differential OpAmp with and without parasitic modeling information

4.6.3 A Compaction Example

Once the parasitics and performances of the optimized design are formally extracted and validated, it can be fine-tuned with the compaction procedure. Firstly, the routing of layout in Fig. 4–21(a) is removed, leaving with the block placement and its geometric constraints. Fig. 4–21(b) and 4–21(c) illustrate the MHS partition and the derived placement constraints of an operational amplifier layout. Device cores, including routing area, are represented by rectangular blocks such that they allow flushed packing without design rule violations. A cost function formed by the linear combination of substrate and interconnect parasitics is used to permit optimization via Simulated Annealing. The compaction configuration with the least parasitic cost is obtained, shown in Fig. 4–21(d).

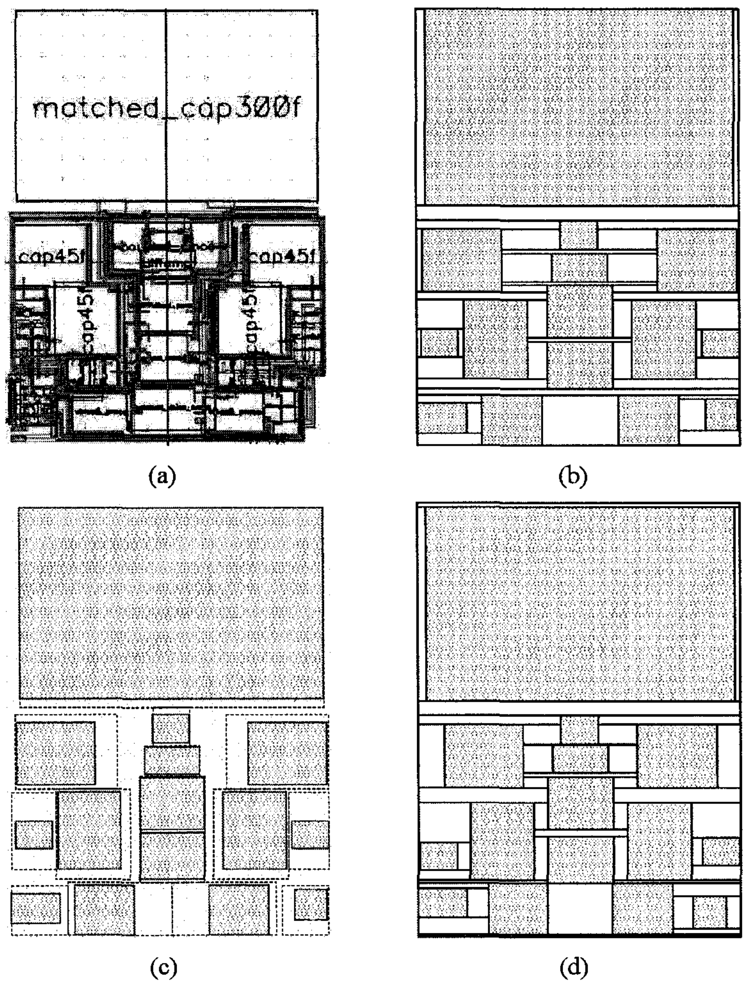
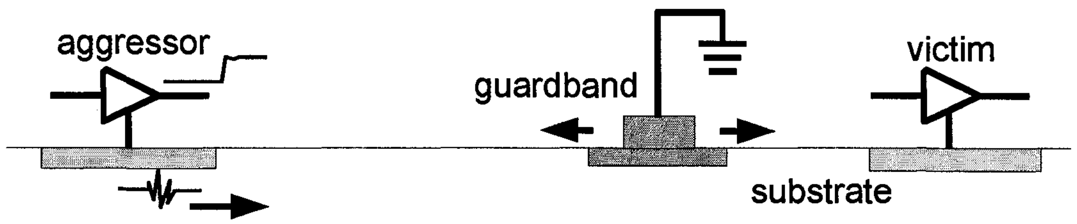
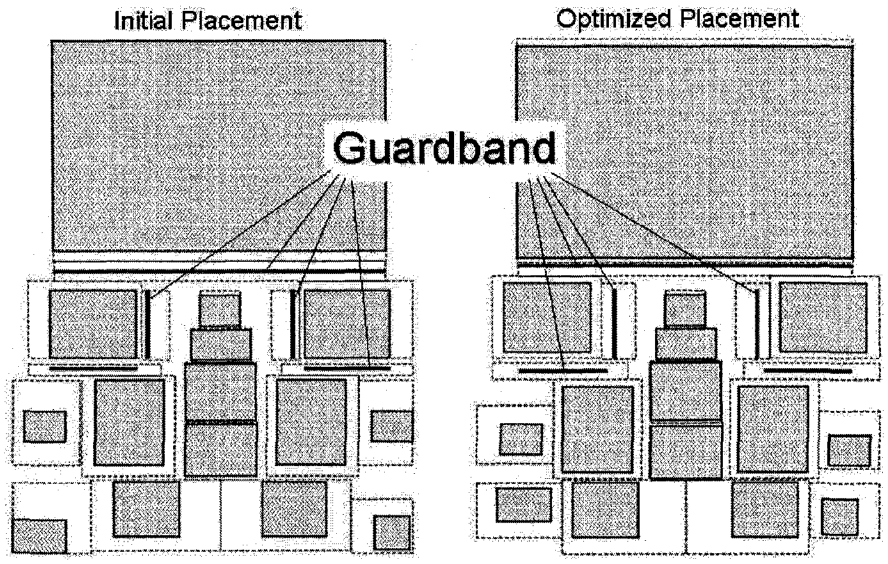


Figure 4–21: (a) Placement extraction from layout (b) MHS partition (c) Compaction constraints (d) Optimized placement for OpAmp compaction



(a) Placement of biased guardband between aggressor and vicitim to minimize substrate coupling



(b) Original and optimized placement on left and right respectively

Figure 4–22: Guardband placement optimization

A novel use of the proposed compaction optimization algorithm is to optimize the placement of guardbands. Insertion of guardrings and guardbands between noisy and quiet devices as in Fig. 4–22(a) are popular and economical ways to reduce substrate coupling [6]. It is, however, unclear whether it is more effective to place the guardband near the aggressors or closer to the victims [67]. In the proposed compaction, guardbands are treated as ordinary device blocks with assigned compaction contraints. Through compaction optimization, as the placement of the guardbands is evaluated, the area trade-offs, and the merit of substrate coupling suppression among the aggressors, guardbands and victims are weighed. Fig. 4–22(b) shows an example of compaction optimization with optimized placement of guardbands.

CHAPTER 5

SIMULTANEOUS SWITCHING NOISE-AWARE PLL JITTER MODEL FOR SYSTEM-ON-CHIP DESIGN AUTOMATION

This chapter demonstrates modeling needs for the design of a low-jitter *phase-locked loop* (PLL) circuit using the proposed layout-aware optimization flow. A methodology to model the chip-level statistical jitter of PLLs in SoCs will be presented. Since PLL jitter is determined by both its circuit design and external noise coupling, which in turn is highly dependent on the placement and layout of the SoC, it is a fine example to illustrate the proposed circuit and physical design co-optimization methodology in the previous chapter. Notwithstandingly, efficient evaluation of transient PLL jitter has long been a major obstacle for designers, and even more critical within the optimization loop. The efficiency issue is addressed in twofold by a novel switching noise coupling model using the *Karhunen-Loève* transformation and a *periodic steady-state* [70] jitter equation.

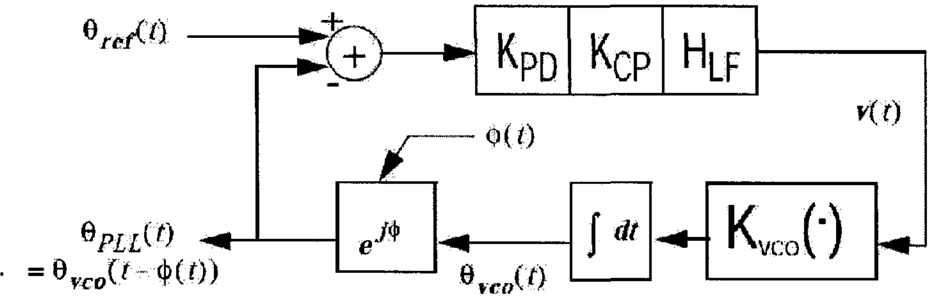


Figure 5–1: Ring oscillator PLL linear model

5.1 Phase-Locked Loops in SoCs

SoCs integrate the entire system onto a common silicon die for the reduction of production cost, form factor and decrease communication distances. Function blocks are integrated in close proximity to share the power supply and substrate. Among these tightly-integrated mixed-signal systems, power supply noise and substrate coupling are the prevalent external interference [2]. They arise from the coupling of impulsive charge injection generated collectively by large number of digital gates switching synchronously, hence commonly known as *simultaneous switching noise* (SSN). Increased device density, packaging parasitics and low substrate resistivity deteriorate the electrical isolation among subsystems. Moreover, the bulk CMOS fabrication process almost exclusively used to implement SoCs does not provide outstanding insulations. For instance, average transient peaks of global chip substrate potentials in excess of 0.1V are common for million-transistor digital circuits.

PLL is a crucial and versatile component in the SoC architecture for applications such as synthesis of multiple clock frequencies for *clock domains*, and clock and data recovery

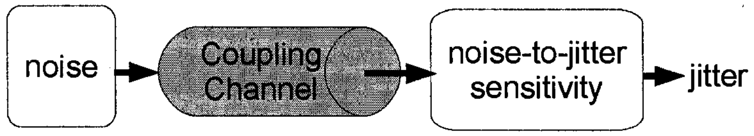


Figure 5–2: Simulation setup of SSN-induced PLL jitter

at high-speed communication interfaces. Due to the increase in the number of clock domains and communication interfaces in SoCs, multiple PLLs are needed. The quality of each PLL is vital to ensure proper operation of the systems it serves. This can be measured by the range of lockable frequencies and the precision of its oscillation frequency, with the latter measured by the time interval between threshold-crossings of its oscillating output, assuming an ideal reference frequency source. The short-term variations of this interval from its mean value are known as *timing jitter*, or simply jitter. The jitter of a PLL output clock signal can be affected by its circuit design, and the random jitter of the oscillator component. Noise can emanate from within the PLL or external sources. The noise response block diagram of a generic PLL is shown in Fig. 5–1. The sources of noise entering the PLL are the intrinsic noise from its own devices or interference from tightly integrated current drivers and digital gates, through substrate coupling, or other parasitic means in mixed-signal system-on-chips (SoCs). Accurate estimation of chip-level SSN coupling and parasitic effects are thus needed. The rule of thumb for selection of transient simulation time step is $1/50$th of the period of the maximum frequency of interest. To simulate jitter, a long duration of relatively small transient time steps is required due to its small value in the neighborhood of pico-seconds. In this chapter, a jitter analysis technique based on *periodic steady-state* (PSS) analysis will be presented.

The total jitter of PLLs in SoCs are assumed to be the sum of its instrinsic jitter and *simultaneous switching noise* (SSN)-induced jitter. The first component is obtained by following traditional steady-state jitter analysis using well-known linear time-invariant PLL noise models [29]. Meanwhile, to compute the second part, a *periodic steady-state* (PSS) analysis of SSN in the SoC has to be performed. SSN can enter the PLL are through the interconnects, power supply, and substrate coupling due to device leakages. They share a common simulation setup, which consists of obtaining the specific switching noise profile depicted in Fig. 5–2. It consists of obtaining the following entities:

1. SSN sources,
2. Parasitic values of coupling channels, and
3. Noise sensitivity of target circuitry.

The first obstacle to be overcome is the sheer number of gates needed to be excited at particular states of the large system in order to generate realistic SSN. The next few sections describe how SoC subsystem blocks of various clock domains are replaced by SSN models that capture their noise signatures, via the *Karhunen-Loève* expansion [38] method. The parasitic coupling paths leading to the PLL are extracted next. Finally, the noise sensitivity of the PLL is used via PSS to compute the jitter induced by SSN.

5.2 Efficient Modeling of Chip-Level Simultaneous Switching Noise

SSN is a general description of all parasitic couplings that are related to switching activities. Major SSN contributors are large number of high-activity switching devices, such as memory and gate arrays, and signal processors, or those with large current swings, such as input and output pin drivers and nearby switched capacitors. They can be quantified by their supply currents. Due to aggressive SoC integration, complete isolation of PLLs from mixed-signal noise is impractical, hence maintaining PLL performance must rely on quality circuit design practice and accurate predictions of SSN.

SSN distinguishes itself from *stationary* intrinsic device noise by their *temporal correlations* with clock signals. It is a *cyclostationary* stochastic process, versus the stationary process of the device noise types. The specific combination of the states of the actively switching circuits dictate the characteristics of SSN signatures. Since clock sources are usually driven by nearby PLLs, the periodicities and power spectra are thus related to the PLL power spectrum. This temporal correlation may render SSN more detrimental to PLL performance than other noise, since the ambient noise level constantly changes at rates much shorter than the response time of the PLL feedback control loop. SSN affects normal system operation by varying delay characteristics and junction potentials of transistors.

The following proposes a compact SSN model that characterizes it noise signatures from data samples. The coupling path parasitics and the VCO noise sensitivity will be derived in Section 5.4.

5.2.1 Cyclostationarity

SSN is stochastic, periodic and with the majority of activities concentrated in certain short intervals relative to the clock transitions. An example of the substrate coupled SSN observed at the bulk substrate beneath the VCO is shown in Fig. 5–3. In clock distribution networks, this switching noise source is highly related to the VCO transitions. The SSN phenomenon is modeled as a *wide-sense cyclostationary* process, or simply cyclostationary process, which means that it is a *Gaussian* process with *mean* and *variance* vary

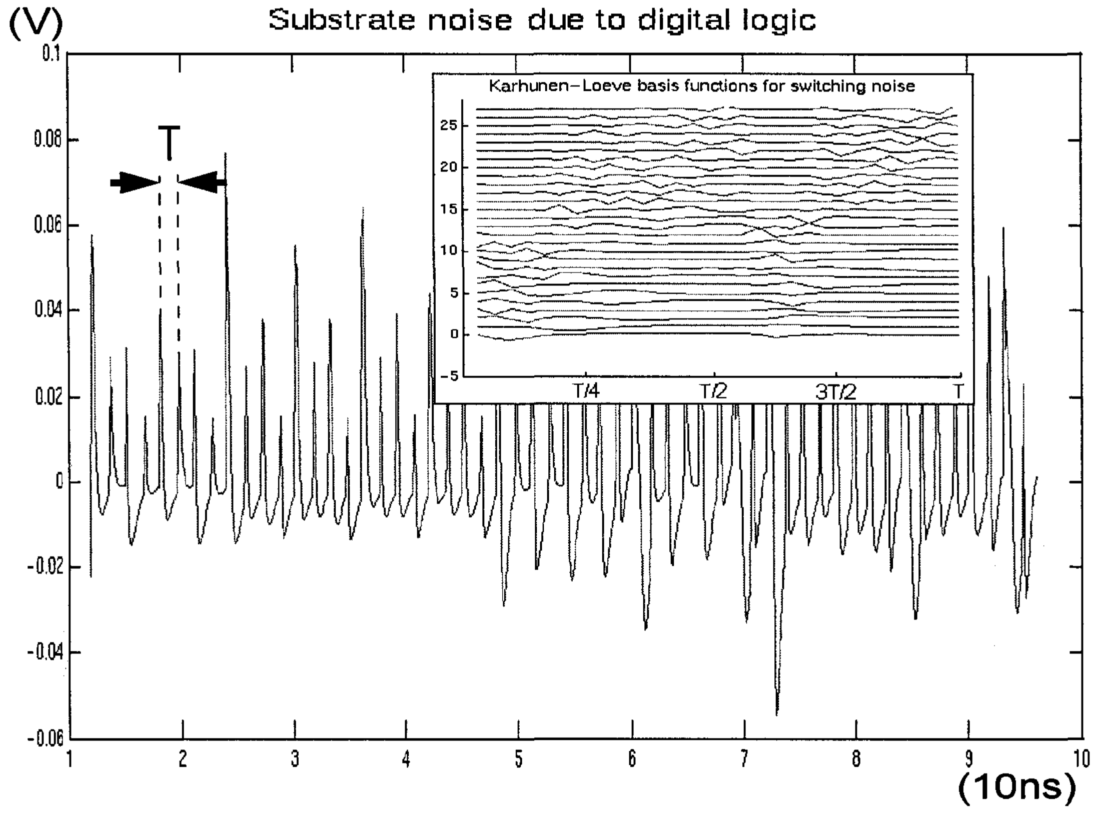


Figure 5–3: Substrate coupling due to digital logic switching activities and its KL basis functions (inset figure)

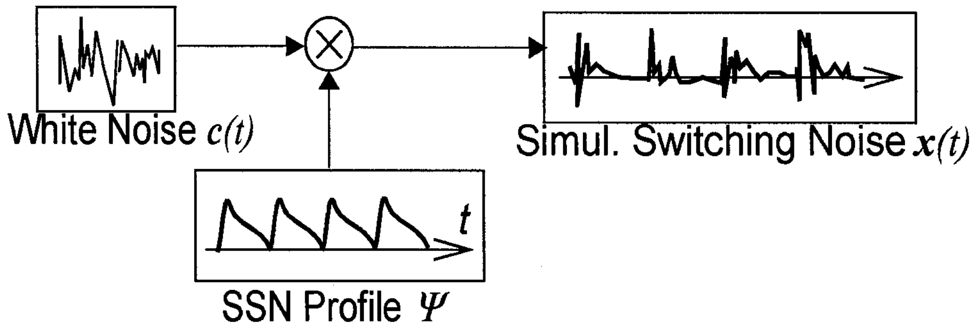


Figure 5–4: Equivalent model of SSN generation from noise profiles derived from substrate coupling simulations

periodically with time [4]. To obtain accurate SSN waveforms without costly simulation of large circuits, this work proposes a compact cyclostationary model that captures the distinctive noise profile $f(t)$ from SSN samples derived from simulations, where complete periods of SSN samples are weighted by their state occurrence probabilities. Suppose a SSN period spans P simulation time steps, the i^{th} period of model-generated SSN x is given by

$$[x_{i1}, ..., x_{iP}]^t = \sum_{j=1}^{N} c_{ij}\psi_j(t) = \Psi \mathbf{c_i} \tag{5.1}$$

$\mathbf{c_i} = [x_{i1}, ..., x_{iN}]^t$ is a column vector of *uncorrelated* normally-distributed coefficients. (5.1) implies that $x(t)$ is generated periodically by $f \equiv \Psi$ and N *stationary* Gaussian weights $\mathbf{c_i}$ as depicted in Fig. 5–4. The rule of thumb of determine the simulation time step is about 50 times the frequency of interest [reference needed]. Hence, a high temporal resolution for SSN is crucial. Close examination of (5.1) implies that reducing the time step of SSN to improve its resolution will also increase the number of basis functions $\psi_j(t)$ the model needed. Consequently, the computational complexity of SSN generation also increases at the same rate. Jitter evaluation efficiency could be significantly improved if the quality of the SSN model can be de-coupled from its computational complexity, such that high quality noise signature can be represented by a low-order model. This is achieved through an algebraic transformation called the *Karhunen-Loève* (KL) transformation. The following derive the cyclostationary basis Ψ in (5.1) and apply the KL transformation to it.

5.2.2 The Karhunen-Loève Basis

The SSN basis is determined adaptively from data features. More specifically, it is the *eigenspace* of the *autocorrelation matrix* of the data samples. Basis Ψ in (5.1) is a $P \times N$ matrix $[\psi_1(t), ..., \psi_N(t)]$ denoting the column space of a set of basis functions. It suffices for to be of full column rank, i.e. $N \leq P$. Detailed derivation of the optimum basis functions using the Karhunen-Loève (KL) method is shown in Section 5.6.1 in the Appendix. Write the autocorrelation matrix

$$R_{xx} = \sum_{i=1}^{M} E[x_i x_i^t] = \Psi \sum_{i=1}^{M} E[c_i c_i^t] \Psi^t = \Psi diag(\lambda) \Psi^t \Rightarrow R_{xx}\Psi = \Psi diag(\lambda) \quad (5.2)$$

where $\Psi = \{\psi_j\}$ are eigenvectors of R_{xx}:

$$R_{xx}\psi_{\mathrm{j}} = \lambda_j \psi_{\mathrm{j}} \quad (5.3)$$

The eigenvalue decomposition essentially detects correlations among multiple variables in the feature space x, de-correlate them, and map them to the transform space c, depicted in Fig. 5–5.

It has been proven that the KL transformation optimally reduces the model order with minimal loss in accuracy.

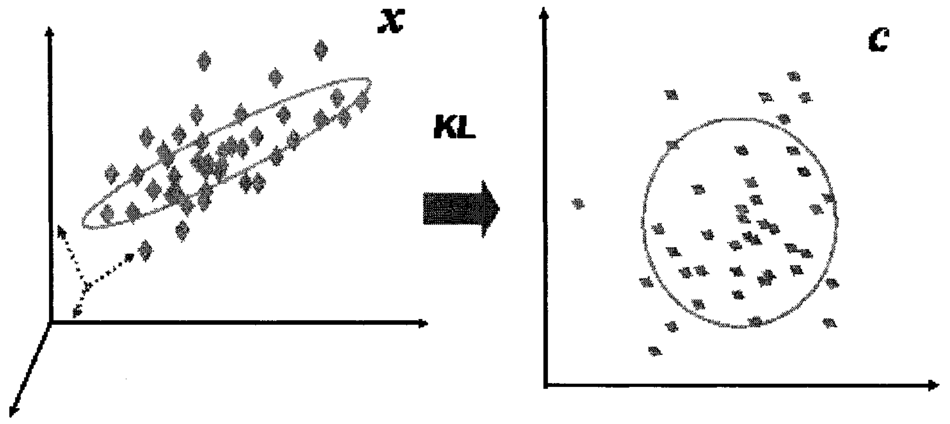


Figure 5–5: Dimensional reduction and removal of variable correlations from *feature space* x to the *transform space* c through KL mapping

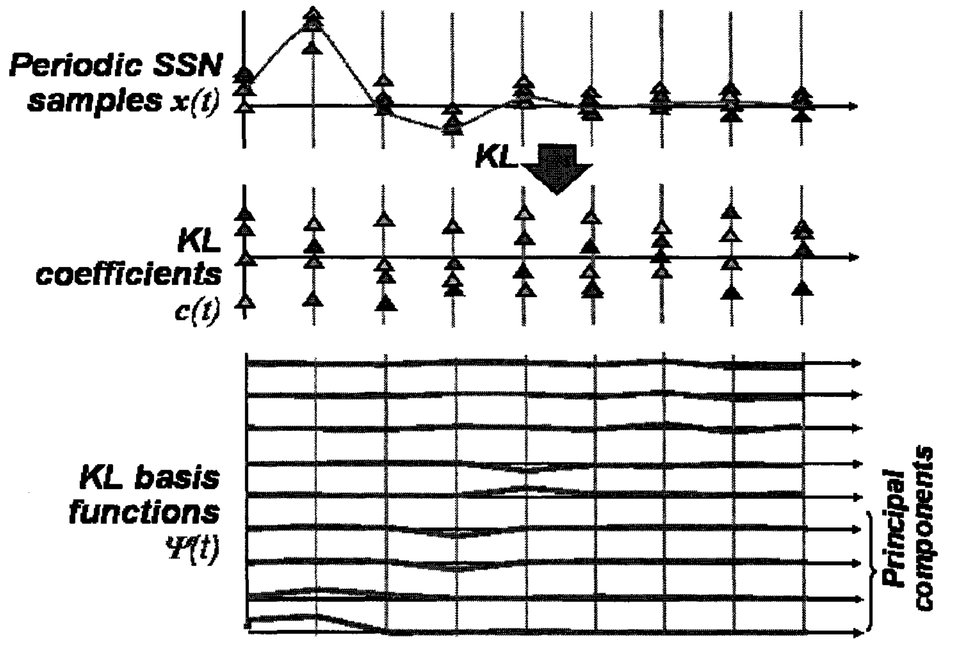


Figure 5–6: Example of KL basis extraction and its corresponding distribution of coefficients from data samples

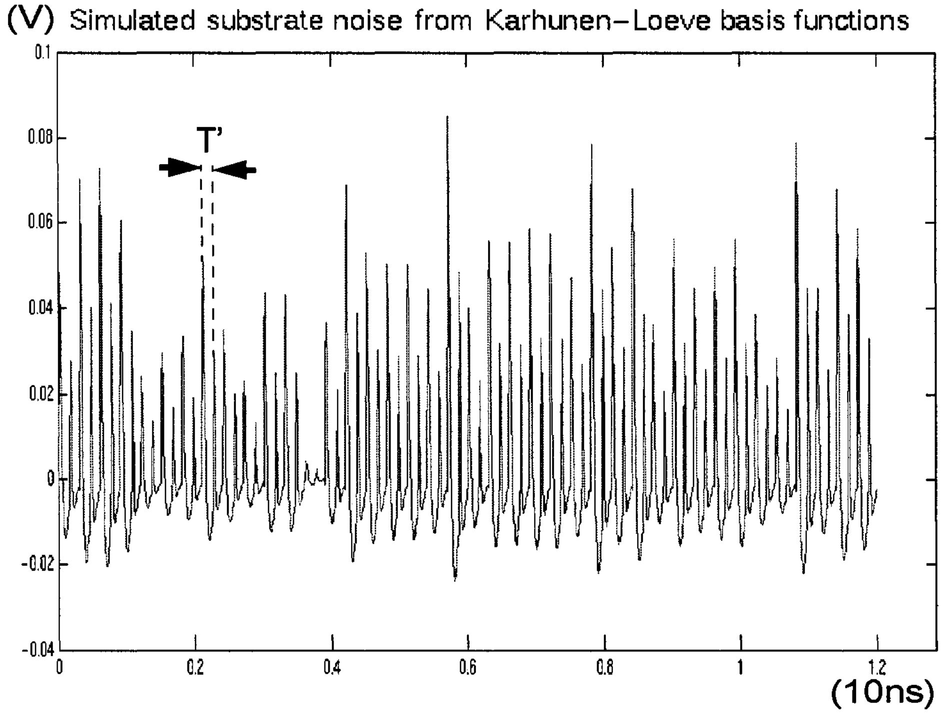


Figure 5–7: KL model-generated new noise samples

Given SSN profiles from various subsystem blocks, distinctive KL basis functions are extracted to represent their individual SSN signatures. Examples of the KL basis derived from SSN samples are shown in Fig. 5–6, as well as the inset figure in Fig. 5–3. A fundamental property of KL expansion is that it is the optimal linear transformation using orthogonal basis functions that maximize the variances of the dataset when projected on them. For partial basis expansions in particular, it aligns them along the data in the most interesting dimensions. Only a few basis functions are often needed to adequately capture any given SSN signature, providing a compact and high-resolution model.

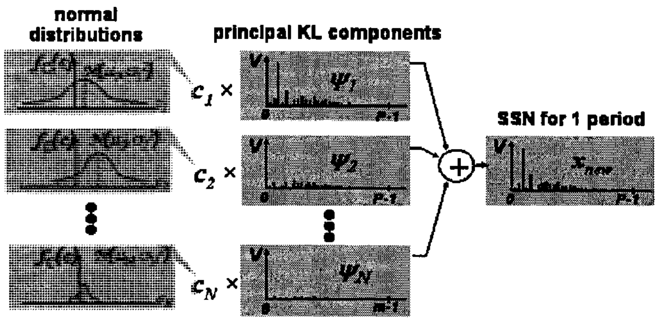


Figure 5–8: SSN generation from reduced-order SSN model

If the basis functions are sorted with the largest corresponding eigenvalues first, this ordered KL expansion gives the most accurate representation for a reduced number (usually < 10) of basis functions by retaining the first few *principal* components and truncate the rest. Using $N \ll P$ principal components in (5.1) reproduces a periodic SSN sequence at minimal computation cost

$$x_{new}(t) = x(t' + kT_f) = \mathbf{f}(t')\mathbf{c}, \quad t' \in [0, T_f). \tag{5.4}$$

where k is an integer, $\mathbf{f}(t') = [\psi_1(t'), \psi_2(t'), ..., \psi_N(t')]$, and $\mathbf{c} = [c_1, c_2, ..., c_N]^t$. $c_1, c_2, ...$ are random scalar coefficients drawn from its corresponding distribution every period. The new SSN waveform can be easily scaled to investigate its relation to the PLL frequency. An example of $x_{new}(t)$ using 10 basis functions is shown in Fig. 5–7.

5.3 Modeling the SSN Coupling Paths

Parasitic coupling paths for each SSN source are established via a parasitic extraction scheme. Each $x_i(t)$ and $h_i(t)$ in Fig. 5–9 correspond respectively to a SSN source and the impulse response of its coupling path, contribute to the SSN perceived by the VCO.

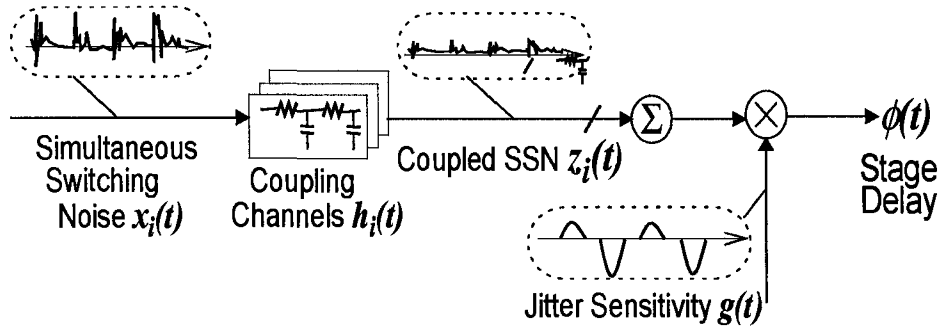

Simultaneous Switching Noise $x_i(t)$
Coupling Channels $h_i(t)$
Coupled SSN $z_i(t)$
Σ
Jitter Sensitivity $g(t)$
$\phi(t)$
Stage Delay

Figure 5–9: Noise-to-jitter transfer model

$g(t)$ is its *jitter sensitivity*. In particular, due to well bias junction capacitances, *p*MOS in *n*-wells are more insulated from SSN coupling than the *n*MOS in the bulk substrate. As SoC performance is sensitive to interconnect coupling, particularly noisy SSN sources and high conductivity paths must be identified as early as possible in the design flow.

Next, the transfer function $H(j\omega)$ of the parasitic coupling paths is computed by the impulse responses $h_i(t)$ obtained through post-extraction netlist simulations, where multiple coupling paths from various clock domains are extracted from layout. In cases where exact substrate or routing details are not available, coupling transfer functions are estimated based on the proposed substrate coupling model and virtual interconnect paths. They accounted for material properties, block separation distances and parasitic conductivities. For substrate coupling, epitaxial and bulk conductivities and geometric distances among the geometric centroids of the digital blocks and the VCO are used. *n*-well devices body connections can also be derived from their direct distances to their doping tubs bias nodes. Coupling across the power supply nets can be modeled the similarly based on wire length.

The following section explores the effects of the relative phase and frequency differences among $x_i(t)$ and $g(t)$. In the following development of the jitter equation, functions are represented in the continuous time domain to avoid the additional confusions of the multiple sampling frequencies for each time function. The discrete time equivalence of the final equation will given in the numerical implementation section.

5.4 PLL Jitter Analysis

A typical integrated PLL consists of a *voltage-controlled oscillator* (VCO) and a feedback control circuit. Due to the ease of integration and low-Q, low gain limitations, *ring oscillator* (RO) VCOs are the preferred implementation in CMOS SoCs over tuned LC circuits and relaxation oscillators. The VCO oscillation frequency is dictated by a control voltage, regulated by the control loop. It acts to compensate for the phase differences between the PLL output and a reference signal. In its steady-state locked to a reference signal, an ideal PLL output signal can be represented by a periodic voltage $v(t)$. In practice, the steady-state mean-square jitter of a PLL waveform is non-zero. Its Fourier series expansion can thus be expressed as

$$v(t) = \sum_{n=-\infty}^{\infty} A_n exp(jn(\frac{2\pi t}{T_0} - \phi(t))) \tag{5.5}$$

where $\phi(t)$ denotes the *phase lag*, or *phase noise*, due to interference and non-idealities, accumulated from all previous oscillation cycles. Fig. 5–1 shows the propagation of $\psi(t)$ in a linear PLL model. Even when $v(t)$ is relatively noise-free, $\phi(t)$ is nonzero and monotonically non-decreasing until it is sensed by the Loop Filter (LF) and eventually corrected by altering $v(t)$. The loop filter is a low-pass filter that has a lower bandwidth than the

VCO (typically $1/20^{th}$ to $1/100^{th}$). During the initial tens of cycles, $\phi(t)$ is passed to the PLL output unregulated by the feedback loop. For instance, suppose a CDR circuit driven by a PLL is interfered by SSN, with comparable periodicity as the PLL. As the PLL locks onto the phase of the incoming data signal, its transition is skewed by SSN, creating jitter. Due to the low-pass nature of LF, the feedback mechanism typically requires the order of a few tens of cycles to compensate for the jitter. At the next PLL cycle, a new group of SSN pulses of *different* intensity arrives before this jitter is effectively eliminated. This cycle-to-cycle statistical variations renders the feedback mechanism inefficient. As long as the SSN fundamental frequency is much less than the LF bandwidth, short-term SSN-induced PLL jitter is largely determined by the jitter in the VCO, or $\phi(t) \approx \phi_{vco}(t)$. Let t_2 be the elapse time for $v(t)$ in (5.5) to complete N VCO periods T_0 since time t_1, or $t_2 - t_1 = NT_0 + \Delta t$. Δt is called the *timing jitter*. It can be expressed in terms of absolute time or per *unit interval* (UI) [1]. Factors contributing to $\phi_{vco}(t)$ arise from *intrinsic* circuit device noise, such as thermal, shot, flicker noise, or from various *external* sources, such as signal line crosstalk, power supply noise, well and substrate coupling. From (5.5),

$$\begin{cases} 2\pi t_1/T_0 - \phi(t_1) = t_0 \\ 2\pi t_2/T_0 - \phi(t_2) = t_0 + 2\pi N \end{cases} \tag{5.6}$$

Taking the difference of the equation pair, the total timing jitter between time t_1 and t_2 is

$$\Delta t(t_1, t_2) = \frac{T_0}{2\pi}(\phi_2 - \phi_1) \tag{5.7}$$

where $\phi_i = \phi(t_i)$. The mean square timing jitter is

$$\langle \Delta t^2(t_1, t_2) \rangle = \frac{T_0^2}{4\pi^2}(R_{\phi_1\phi_1} + R_{\phi_2\phi_2} - 2R_{\phi_1\phi_2}) \tag{5.8}$$

where R is the *autocorrelation* function of $\phi(i)$. Suppose intrinsic and external noise components are additive, suppose that $\phi(i)$ is contributed by *stationary* and *cyclostationary* noise sources in SoC environments. Substantial research effort [11],[29] have been devoted to analyze jitter due to the first type. Compute the jitter component induced by the cyclostationary noise in SoCs and refer the *SSN-induced jitter* component simply as *jitter* from this point on unless stated otherwise.

5.5 Computation of SSN-Induced Jitter

In clock generation and synchronization applications, the number of delay elements in the ring and their individual propagation delays are important design parameters affecting its precision and operable frequency range. Suppose an oscillator has a ring connection of n delay elements. The *instantaneous phase lag* of one infected delay element (together with $n-1$ noiseless elements) due to SSN coupling as

$$\varphi(t) = [h(t) * x(t)]g(t) = \int_{-\infty}^{t} h(t-\tau)\mathrm{f}(\tau)\mathrm{c}d\tau \cdot g(t) \tag{5.9}$$

$g(t)$ is the time-varying noise sensitivity of one stage of the ring oscillator (RO). It is periodic with oscillator period T_g and thus has discrete Fourier expansion $\sum_{n=-\infty}^{\infty} G_n exp(j2\pi nt/T_g)$. Assuming an N-stage inverter (differential amplifier) RO, sharing identical delay and parasitic characteristics, there are n falling and n rising transitions in each period. For CMOS inverters, the falling and rising transitions are sensitive to substrate and n-well coupled noise respectively, while for differential amplifiers, both transitions types are sensitive to coupled SSN. The instantaneous jitter model for differential amplifiers RO is shown in Fig. 5–10. The coupled noise $z(t)$ is sampled at N evenly-distributed instances per period.

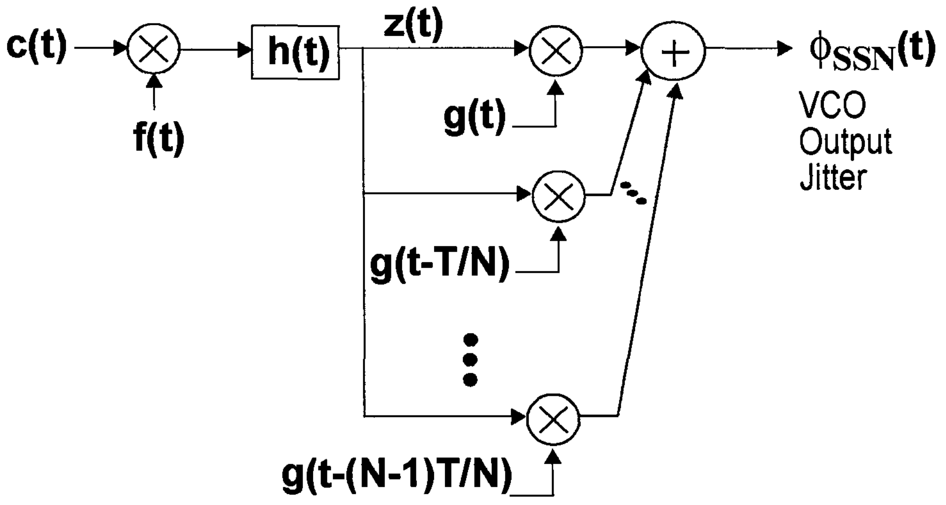


Figure 5–10: N-stage ring oscillator output jitter model

The *instantaneous* jitter due to SSN is thus of the N time-shifted sensitivity functions super-imposed

$$\varphi(t) = \int_{-\infty}^{t} h(t-\tau)\mathrm{f}(\tau)\mathrm{c}d\tau \cdot \sum_{l=0}^{N-1} g(t - \frac{lT_g}{N}) \tag{5.10}$$

To explore the temporal correlations between the SSN and PLL noise sensitivity, expand the spectra of $\mathrm{f}(t)$ and $g(t)$. $x(t)$ is projected onto its partial KL basis in (5.4). To show its interactions with $g(t)$, (5.4) is projected onto the complex exponential basis

$$x(t) = \mathrm{e}^t\Psi\mathrm{c} = \sum_{m=-\infty}^{\infty}\sum_{i=1}^{N} \Psi_{im}c_i\exp(\frac{j2\pi m\tau}{T_f}) \tag{5.11}$$

where $\Psi_{im} = 1/T_f \int_{T_f} \Psi_i(\tau)\exp(j2\pi m\tau/T_f)d\tau$, T_f is the SSN period. Let $\mathrm{F}_m = [\Psi_{1m}, \Psi_{2m}, ..., \Psi_{Nm}]$, the instantaneous phase lag becomes

$$\varphi(t) = \sum_{m,r=-\infty}^{\infty} A_{m,r}\exp(j2\pi t(\frac{rN}{T_g} + \frac{m}{T_f})) \tag{5.12}$$

where $A_{m,r} = NH_m\mathbf{F}_m\mathrm{c}G_{rN}$. Refer to Appendix section for derivation details of (5.12). The phase noise at the i_{th} transition is the integral of $\varphi(t)$ to the current transition time

$$\phi[i] = \phi(iT_g) = \int_0^{iT_g} \varphi(t)dt, \text{ while } \phi[i]lliT_g \tag{5.13}$$

For the most accurate treatment, the end time of integral (5.14) should be taken as exact current time with jitter, i.e. $iT_g + \phi[i]$. However, $\phi[i]$ has to be solved at each time step iteratively. Instead, for $\phi[i] \ll iT_g$, (5.14) is simplified using the periodic steady state assumption, to integrate up to iT_g in order to give an approximated analytical solution.

In practice, the periodic switching noise $f(t)$ and PLL sensitivity profile g(t) are obtained through numerical simulations with time step t_s. Let $Q = T_g/t_s$, then (5.14) becomes

$$\phi[i] = \sum_{u=0}^{iQ-1} \varphi(ut_s) \tag{5.14}$$

The *root mean square* (RMS) timing jitter at the i_{th} transition is

$$\sqrt{\langle \Delta t^2(0, iT_g)\rangle} = \frac{T_g}{2\pi} \sum_{u=0}^{iQ-1} \sum_{r=0}^{\lfloor \frac{Q-1}{N} \rfloor} \sum_{m=0}^{P-1} \mathrm{A}'_{m,r}\xi_{m,r}(u)\sqrt{\sigma_\mathrm{c}^2} \tag{5.15}$$

where $\mathrm{A}'_{m,r} = NH_m\mathbf{F}_mG_rN$, $\xi_{m,r}(i) = \frac{1-e^{j2\pi miQ/P}}{1-e^{j2\pi(rN/Q+m/P)}}$ and σ_c^2 is the variance of Gaussian random variable c. Refer to Section 7.2 in the Appendix for the derivation details of (5.15). In particular, the term $exp(j2\pi mkQ/P)$ reflects the harmonic interactions between the VCO and SSN. (5.15) provides an efficient analytical model for statistical analysis over various coupling parasitics, spectra of SSN and PLL jitter sensitivities.

5.6 Implementation and Experimental Results

This section shows the performance comparisons of computing VCO jitter affected by cyclostationary SSN interference using the proposed model versus circuit simulation. Application of the SSN model and jitter estimation for SSN-aware PLL design exploration will also be shown, as well as the extraction of the KL basis for the sampled SSN data.

5.6.1 Karhunen-Loève Basis Functions Derivation

Suppose a transient SSN waveform $x(t)$ is given for M clock periods T, and let there be P fixed time steps in each period. As shown in Fig. 5–11(a), $x(t)$ can then be written as a sequence of M column vectors $\mathbf{x}_i(t)$, $0 \le i \le M - q$, $iT \le t \le (i+1)T$, each denotes the SSN sequence for 1 period. Consider its $P \times P$ autocorrelation matrix

$$R_{xx} = \sum_{i=1}^{M} E[\mathbf{x}_i \mathbf{x}_i^t] \, , \tag{5.16}$$

Since the KL expansion for the SSN samples is

$$\mathbf{x}_i = \sum_{j=1}^{P} c_{ij}\psi_j = \Psi \mathbf{c}_i \, , \tag{5.17}$$

as illustrated in Fig. 5–11(b). Thus

$$R = \Psi \text{diag}(\lambda_1, ..., \lambda_P)\Psi^t \tag{5.18}$$

where $\lambda_i = \sum_{i=1}^{M} E[\mathbf{c}_i \mathbf{c}_i^t]$, or

$$R\psi_j = \lambda_i \psi_j \, . \tag{5.19}$$

Hence KL expansion basis can be found by solving for the eigenvalues of the M-averaged autocorrelation matrix R:

1. Form R from SSN samples $\{\mathbf{x}_1(t), ..., \mathbf{x}_M(t)\}$.
2. Obtain eigenvalues $(\lambda_1, ..., \lambda_P)$ of R.
3. Sort eigenvalues $|\lambda_1| \leq |\lambda_2| \leq ... \leq |\lambda_P|$ by their magnitude.
4. Solve $(R - I\lambda_j)\phi_j = 0$ for ϕ_j, $j = 1, ..., P$. I is the identity matrix.

Suppose the estimated *probability density function* (PDF) over M sample periods is Gaussian. Since

$$[\mathbf{x}_1(t), ..., \mathbf{x}_M(t)] = \Psi(t)[\mathbf{c}_1(t), ..., \mathbf{c}_M(t)]\,, \tag{5.20}$$

Compute Gaussian distributions

$$f_{\mathbf{c}_i}(c_j) = \aleph(\mu_j, \sigma_j^2) \tag{5.21}$$

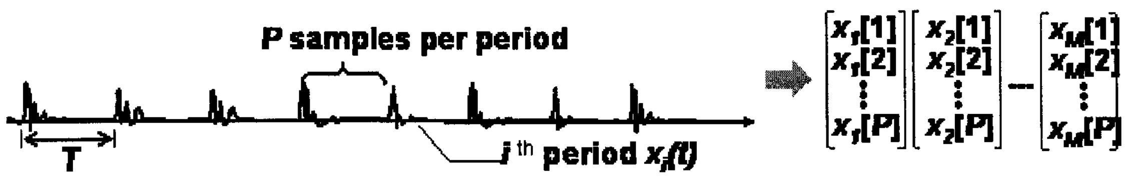
for each KL coefficient c_j, $j = 1, ..., P$ as shown in Fig. 5–11(c), where μ_j and σ_j^2 are the *unbiased* sample mean and variance, respectively,

$$\mu_j = \frac{1}{M}\sum_{i=1}^{M} c_{ij}\,, \text{ and} \tag{5.22}$$

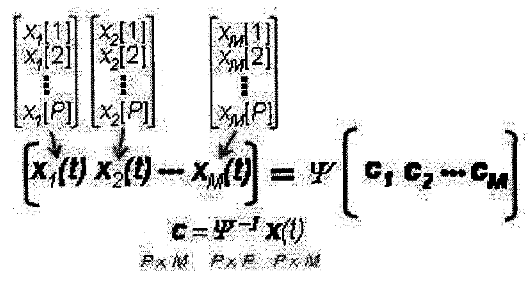
$$\sigma_j^2 = \frac{1}{M-1}\sum_{i=1}^{M}(c_{ij} - \mu_j)^2\,. \tag{5.23}$$

5.6.2 SSN-Aware PLL Design Exploration and Experimental Results

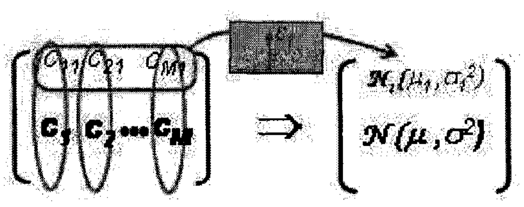
For systems using PLL as frequency synthesizers, it is very probable for the embedded PLL to be interfered by highly-correlated SSN from gate activities that are driven by the same PLL. Moreover, every SoC block has some contributions to every PLL on the same chip. Based on periodic steady-state analysis, the proposed jitter equations (5.12),


P samples per period
T
i th period $x_i(t)$
$x_1[1]$ $x_1[2]$ $x_1[P]$
$x_2[1]$ $x_2[2]$ $x_2[P]$
$x_M[1]$ $x_M[2]$ $x_M[P]$

(a) Arrangement of SSN data samples


$x_1(t)\ x_2(t) \cdots x_M(t)$ = Ψ $c_1\ c_2 \cdots c_M$
$c = \Psi^{-1} x(t)$
$P \times M$ $P \times P$ $P \times M$

(b) Solving for KL basis Ψ


$c_1\ c_2 \cdots c_M$
$\mathcal{N}_1(\mu_1, \sigma_1^2)$
$\mathcal{N}(\mu, \sigma^2)$

(c) Computing statistics of KL coefficients c_i

Figure 5–11: Details of KL-based SSN model construction

(5.14) and (5.15) provide new design insights for SoC PLL circuits and SoC design optimization. The implemented flow shown in Fig. 5–12 uses Cadence SKILL scripts to interface between Cadence DFII and the Python database. Due to the efficiency of the compact SSN models and the jitter equations, the burden of SSN excitation and long PLL simulation time are eliminated. The following effects affecting jitter can be evaluated efficiently:

- PLL design, implementation
- SoC Floor-planning and placement
- Location of PLLs
- Location of Circuit Blocks
- Global on-chip switching activities

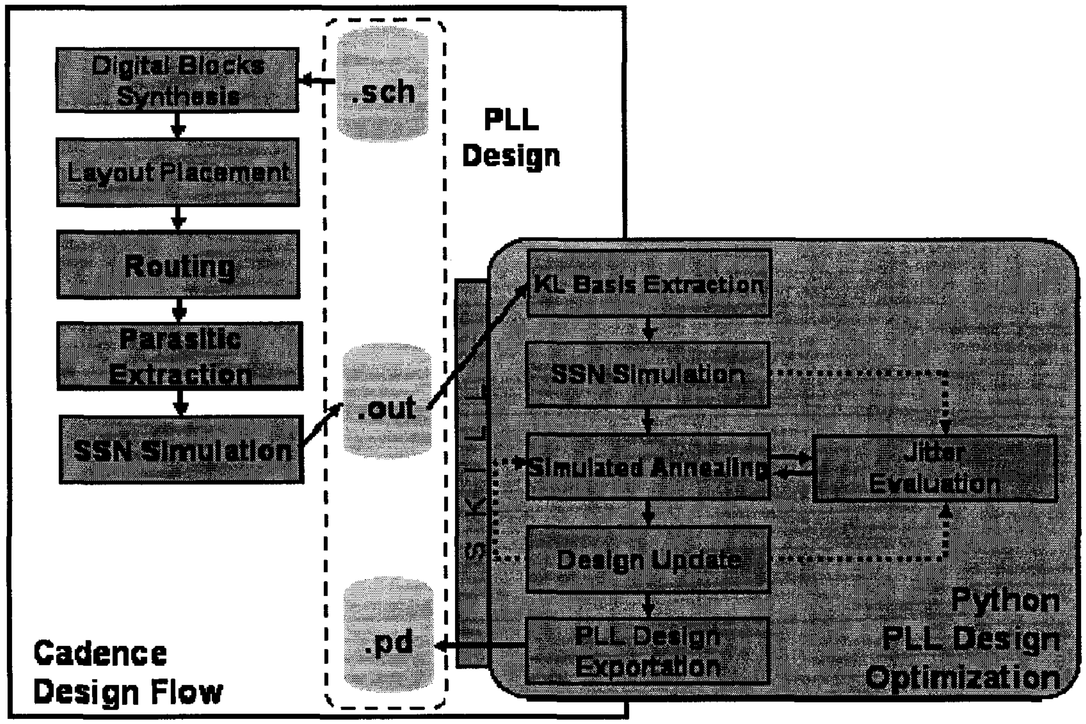
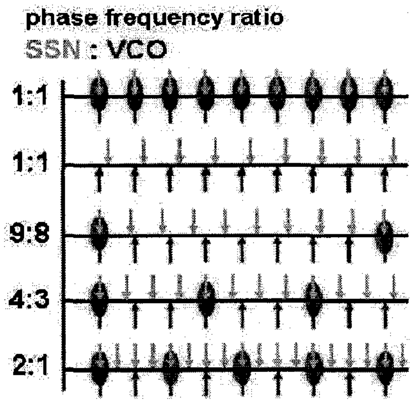
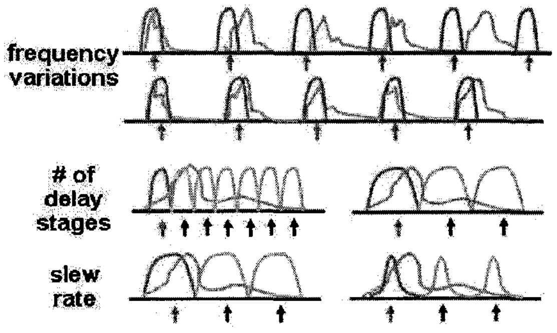


Figure 5–12: PLL circuit and physical design co-optimization



(a) Frequency ratio and phase offset



(b) Effects of VCO (darker color) ring size and frequency ratio

Figure 5–13: Correlations between VCO and SSN in time domain

For example, Fig. 5–13(a) depicts the effect of phase and frequency ratio between the SSN and the VCO sensitivity function. Firstly, the phase plays a significant role in jitter contribution. Assuming for the moment that they are impulses, the first two rows illustrate how a small difference in their relative phase can drastically alter the accumulated jitter. The next few rows illustrate that their frequency ratio also affects the resultant jitter. These patterns imply that there are important implications on the PLL design, such as its frequency range, propagation delay, slew rate and ring size, as illustrated in Fig. 5–13(b). These properties are illustrated below by the experimental results and comparisons.

Practical PLLs are often programmable to use different number of delay stages [14], as well as tuning the delay of each stage [35], this experiment investigates whether there is a jitter performance advantage for using a certain *ring oscillator* (RO) size to implement a given clock signal frequency. As the number of stage n varies, the ambient noise is "sampled" n times at a regular intervals in each VCO period. Cyclostationary noise such as SSN with frequency relationship with the VCO may affect specific oscillation frequencies more than others. Fig. 5–14 compares the number of delay elements in a RO towards its jitter performance using differential stages with different ring sizes (3, 5, 7, 9, 11, 13). The propagation delay for each stage is adjusted such that the RO VCO output signal frequency is unchanged for different ring sizes. Following Fig. 5–10, stochastic equation (5.14) is computed for 50 trials, to obtain representative results. The upper plot shows the accumulated RMS jitter for each trial after 30 oscillation cycles. The statistics of these ensembles are shown in the lower plot. The markers denote the mean RMS jitter of the trial results, while the vertical bars indicate the bounds of one standard deviation around the mean value of the trials. Ring size determines shape of the periodic jitter sensitivity

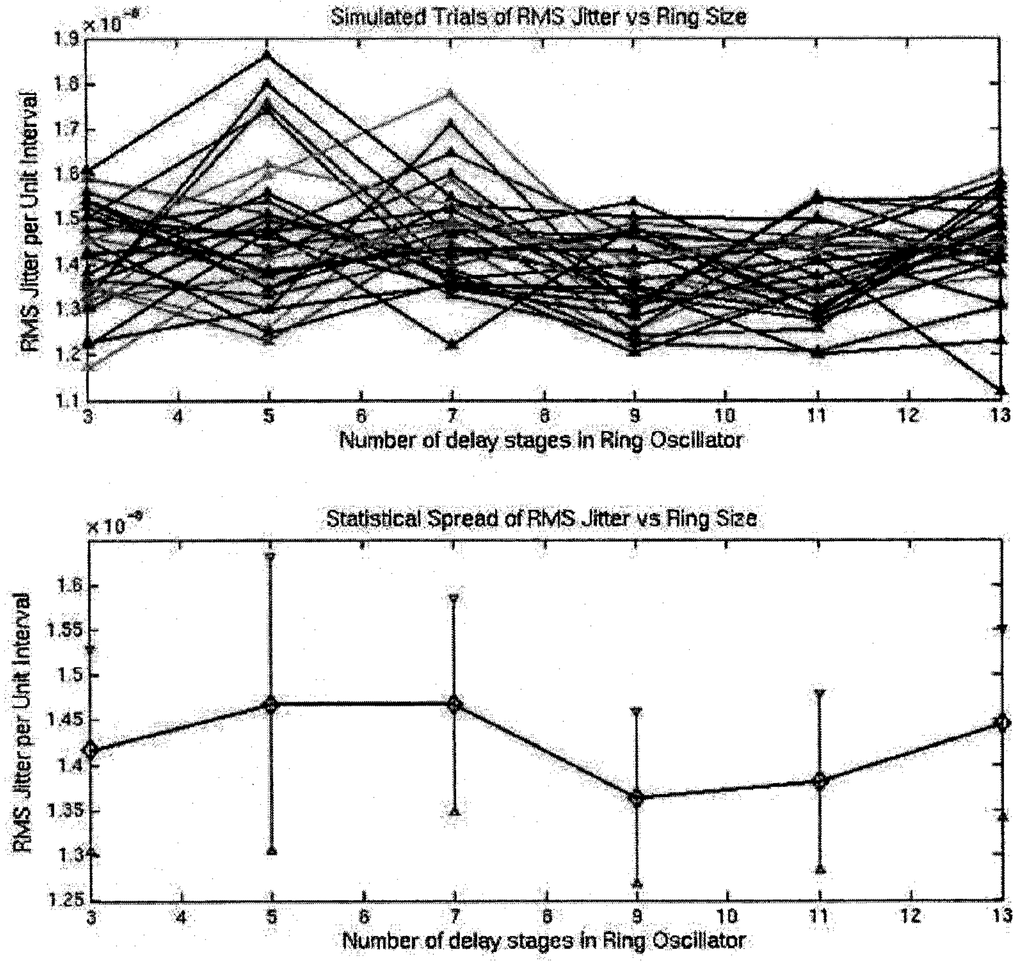


Figure 5–14: RMS jitter with various ring sizes

function of the VCO. In this case, the 9-stage ring results in the best jitter performance in relation to the statistical characteristics of the SSN model. The distinction of jitter performance with identical PLL output frequency, but with different ring size is novel and unique for this jitter analysis technique, which takes noise correlations into account. The jitter performance predicted by conventional white-noise models with identical PLL output frequency would be identical. Via this proposed approach, ring size becomes a new design variable in SoC PLL design.

Similarly, the relative phase among the VCO and the SSN sources also affect the jitter performance. Showing one sweep of a complete VCO period, Fig. 5–15 examines the jitter performance with various timing differences of SSN sources relative to the VCO transitions. The proposed method allows a multi-phase RO PLL to select the best output phase to minimize jitter. On-chip sensors can even dynamically select the relative phase differences of PLL output in response to SSN to minimize jitter.

In the SoC environment, SSN can traversal globally [6] from different parts of the chip to the PLL. The next experiment models SSN interference in SoCs coming from different clock domains. In Fig. 5–16, the jitter due to two SSN sources at the same switching frequency, but are completely in phase and completely out of phase are compared. The result indicates jitter variation depending on the timing offsets among multiple SSN sources, i.e. whether the major noise peaks are distributed throughout the VCO period or concentrated in a small interval. This enables floorplanning and placement of SoCs to consider the relative merits of multiple clock domains distribution, and to be adjusted to minimize their impact.

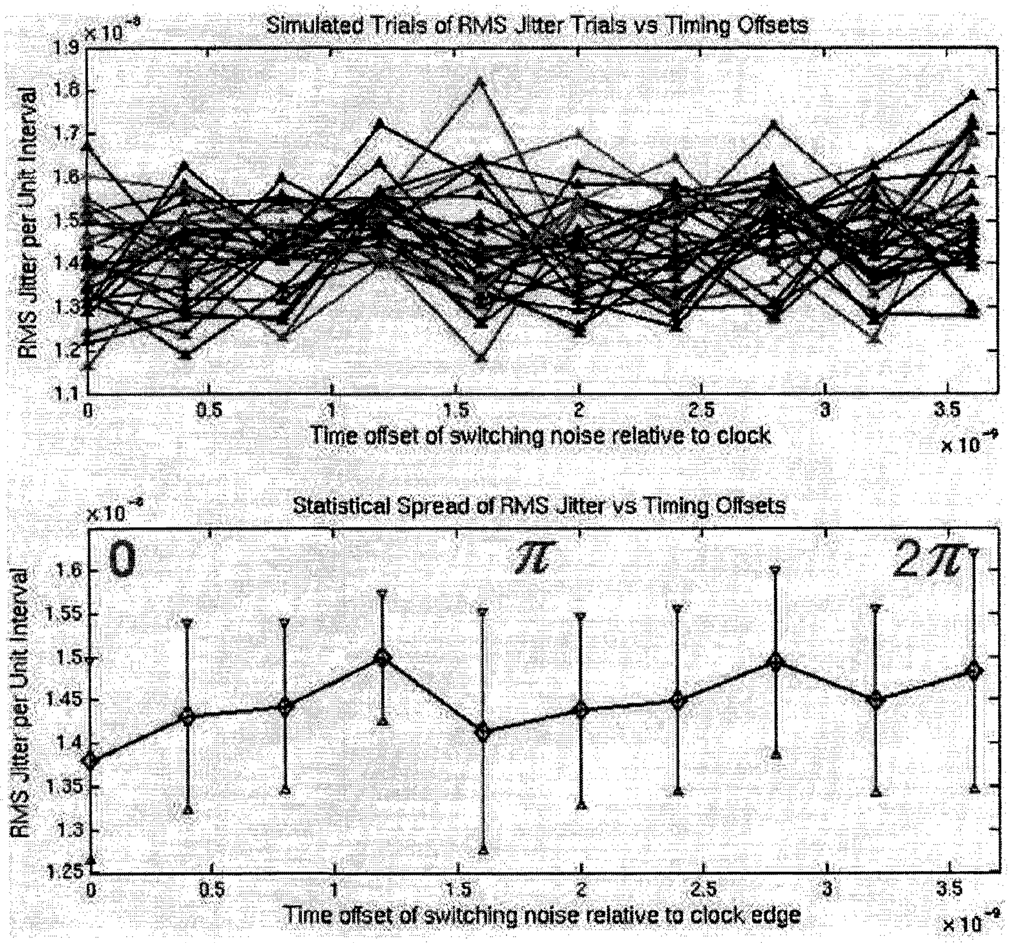


Figure 5–15: RMS jitter with respect to phase offset between VCO and SSN

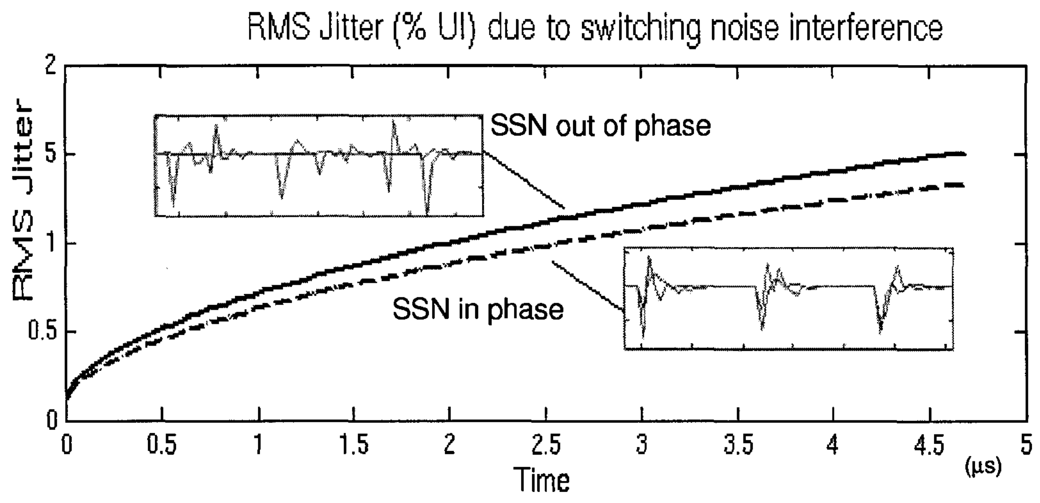


Figure 5–16: RMS jitter with SSN from 2 clock domains with different phase

Table 5–1: Performance comparison between using the proposed jitter model and running Spectre simulator for computing SSN jitter for a 97MHz VCO

VCO Periods	8	16	64	256
Run time (s) (*Proposed Method*)	0.5051	0.5114	0.6086	0.9444
Jitter (ps) (*Proposed Method*)	92.93	97.67	241.5	440.9
Run time (s) (*Spectre*)	153.74	332.8	>30min	>30min
Jitter (ps) (*Spectre*)	119.2	162.4	–	–

Table 5–2: Substrate coupling-induced jitter statistics for 7-stage ring oscillators after 50 periods over 20 trials

DELAY ELEMENT TYPE	FREQUENCY (MHZ)	SLEW RATE (MV/S)	MEAN JITTER (PS)	MEAN RMS JITTER (UI)
CMOS Inverters	237.9	2497	30.98	0.99%
Single-Ended Amplifiers	104.8	587.3	194.3	4.8%
nMOS Differential Amplifiers	156.7	160.9	184.2	4.6%

Table 5–1 compares the efficiency of obtaining jitter performance via SSN models and (5.15), and via the Cadence Spectre circuit simulator. In the setup, an actively switching combinational logic circuit is connected to the VCO through substrate parasitics, and Spectre is configured to use the "moderate" time step control. Only SSN-induced jitter are compared. However, since it is impossible to completely disable instrinsic device noise sources, through the temperature-dependent properties, the temperature parameters `temp` and `tnom` in the device models are set to near 0K during this test in order to suppress the intrinsic noise sources for a fair comparison. Run time of the algorithm is shorter than that of the simulator, especially for long elapsed time. The majority of saved computation time stems from the use of compact noise models and the analytic SSN-induced jitter equation. In Spectre simulation, a 100-gate digital logic block has to be excited every clock cycle. The discrepancies in the jitter quantity can be attributed to numerical error and the stochastic nature of the solutions. Limited time resolution in Spectre analyses, and the linear approximation of jitter sensitivities and other sources of error.

The jitter performance for CMOS inverters (CIs) and differential amplifiers (DAs) implementations with various delay element types are expressed in Table 5–2. Note that for an N-stage RO, there are $2N$ nMOS transitions per period in DAs oscillators and N

nMOS and pMOS transitions for CI oscillators. n-wells are more insulated from SSN coupling than the bulk substrate due to junction capacitance. Hence the CI RO is more robust against jitter *induced by SSN*. Nonetheless, Table 5–2 compares only the theoretical SSN-induced jitter component. Other jitter components [33], [52] are needed to be considered to determine the overall jitter for different RO types.

CHAPTER 6

CONCLUSIONS AND FUTURE DIRECTIONS

The challenges for next-generation analog design automation tools in the DSM era are addressed in this thesis through careful extraction, exchange and reuse of parasitic and performance information between the circuit and layout optimization loops. A co-optimization strategy that allows mutual information exchange between the schematic and placement optimizers has been introduced. Through the sharing of parasitic and performance information, the methodology enables parasitic-aware circuit optimization and performance-driven physical design optimization. A complementary compaction optimizer, a compact chip-level simultaneous switching noise model and a jitter performance equation are also introduced to match the unique properties and requirements of modern analog circuit design.

The issues of parasitic-dominant circuit design automation and the list of original contributions towards solving it were stated in Chapter 1. Basic information about analog circuit design, parasitic effects in the first part of Chapter 2. In the second part, fundamental concepts, implementation of the fundamental components of this work, such as parasitic modeling, representation, estimation of parasitic effects, sensitivity and block placement were discussed in detail. A number of original concepts and implementations described in this chapter laid the groundwork for the implementation of the co-optimization algorithm.

The overall motivations and implementation of particular analog design automation tools are discussed in great details in Chapter 3. Evolutionary, equation-based and simulation-based design automation approaches were visited. They all achieved various success along with specific trade-offs under strong influence of parasitic effects. The novel integration of currently independent circuit and layout optimization flows are proposed in Chapter 4. This enhances the convergence among the optimized circuits and their post-extracted performance at the presence of layout parasitic effects. The overall co-optimization concept was first presented, which consist of five flows accommodating various stages of design completion. Each of these flows were discussed, followed by descriptions of sensitivity-based performance estimation, through geometric, parasitic and performance modifications. Performance-driven compaction optimization and its related components were discussed next, and the implementation and results of co-optimization and compaction were presented towards the end of the chapter. The proposed optimization flow increases the number of available options between two extremes on the accuracy-efficiency trade-off curve, a fast but inaccurate schematic design platform, or an accurate but slow layout design environment. Parasitic estimation was performed based on placement information, and then on both placement and routing information. Later, performance-driven compaction is performed to balance DSM parasitic trade-offs.

The circuit simulator is a highly specialized tool that evaluates practical circuits very efficiently. In certain cases however, simulation can become inefficient. Examples are the transient simulations of PLL jitter and the activities of large number of gate activities. In both cases, transient simulation small time steps that requires solving the circuit system matrices a lot of times repeatedly, or solving very large martices are involved. Frequency domain simulations using sinusoid as signal basis cannot adequately capture enough characteristics in small number of terms, due to the abrupt transitions of mixed-mode signals. Instead, a novel compact SSN model and a jitter performance equation were proposed in Chapter 5. Together, they provide an alternative methodology that permits simulation-based optimization tools to perform PLL design efficiently. The focus on the cyclostationarity of simultaneous switching noise in the proposed method also explores the unique properties of SoC PLL jitter not able to be reproduced with other approaches. Experimental results exhibit the correlations of jitter performance towards oscillator ring size, relative phase and frequency differences with cyclostationary noise.

6.1 Future Work

At present, optimization tools for analog circuits are dedicated to analog circuit designs. A logical extension of the tool is to be able to output parasitic-aware behavioral and sensitivity models of the analog circuits under optimization. These models will then be included with the digital blocks to carry out high-level design optimization and exploration. Another direction yet to be explored in optimization is to apply machine learning techniques in order to better interpret non-ideal performance information. On a more practical note, a sophisticated and consistent user interface will enhance the efficiency and acceptance of analog designers.

Inductance was not included in the proposed work due to the operation frequency range of the majority of analog circuits. Inductance modeling is challenging but becoming critical as signal speed continues to increase above a few GHz.

Finally, automated circuit topology synthesis remains a formidable challenge yet to be solved. There are substantial difficulties in capturing designer experience and knowledge about the implications of each design topology into software. More research in this area is needed. Success on this aspect will represent a major paradigm shift and new methodologies that breakthrough the legacy design flow built around the schematic model.

CHAPTER 7

APPENDIX

7.1 Derivation of Instantaneous Phase Lag Equation (5.12)

We expanded the spectra of $\mathbf{f}(t)$ in (5.4) to obtain (5.11), repeated here as

$$x(t) = \mathbf{e}^t \boldsymbol{\Psi} \mathbf{c} = \sum_{m=-\infty}^{\infty} \sum_{i=1}^{N} \Psi_{im} c_i \exp(\frac{j2\pi m\tau}{T_f}) , \tag{7.1}$$

where $\Psi_{im} = 1/T_f \int_{T_f} \psi_i(\tau) \exp(j2\pi m\tau/T_f) d\tau$, and T_f is the SSN period. Let $\mathbf{F}_m = [\Psi_{1m}, \Psi_{2m}, ..., \Psi_{Nm}]$, the instantaneous phase lag

$$\varphi(t) = \sum_{l=0}^{N-1} \sum_{m,n=-\infty}^{\infty} \mathbf{F_m c} G_n \exp(j2\pi n(\frac{t}{T_g} - \frac{l}{N})) \int_{-\infty}^{t} h(t-\tau) \exp(\frac{j2\pi m\tau}{T_f}) d\tau \, . \tag{7.2}$$

The integral in (7.2) can be written as

$$\exp(\frac{j2\pi mt}{T_f}) \int_0^{\infty} h(\sigma) \exp(\frac{-j2\pi m\sigma}{T_f}) d\sigma \, , \tag{7.3}$$

where $\sigma = t - \tau$, $d\sigma = -d\tau$ and $\tau : (-\infty, t) \to \sigma : (\infty, 0)$. Note that the integral in (7.3) is the unilateral Fourier transform $H(j2\pi m/T_f)$ of $h(t)$. Substituting (7.3) in (7.2),

$$\varphi(t) = \sum_{l=0}^{N-1} \sum_{m,n=-\infty}^{\infty} \mathbf{F_m} c H_m G_n \exp(j2\pi t(\frac{n}{T_g} - \frac{m}{T_f})) \exp(\frac{-j2\pi nl}{N}), \tag{7.4}$$

where $H_m = H(j2\pi m/T_f)$. As

$$\sum_{l=0}^{N-1} \exp(\frac{-j2\pi nl}{N}) = \begin{cases} N & \text{if } n = rN, r \text{ integer} \\ 0 & \text{otherwise} \end{cases}, \tag{7.5}$$

we have

$$\varphi(t) = \sum_{m,r=-\infty}^{\infty} A_{m,r} \exp(j2\pi t(\frac{rN}{T_g} + \frac{m}{T_f})). \tag{7.6}$$

where $A_{m,r} = N H_m \mathbf{F}_m c G_{rN}$.

7.2 Derivation of Mean-Square Phase Noise Equation (5.15)

Assume that both the noise profile f and the VCO sensitivity function g share a common simulation time step t_s, and their discrete time notation $f[i] = f(it_s)$ and $g[i] = g(it_s)$. Their spectra contain *finite* number of Fourier components

$\mathbf{F}_m$, $m = 0, 1, ..., P-1$, and G_n,$n = 0, 1, ..., Q-1$,

where $P = T_f/t_s$ and $Q = T_g/t_s$ are natural numbers. The accumulated phase noise is a partial sum (5.14) of the geometric time series (7.6). Hence

$$\phi[i] = \sum_{u=0}^{iQ-1} \sum_{r=0}^{\lfloor \frac{Q-1}{N} \rfloor} \sum_{m=0}^{P-1} A_{m,r} \xi_{m,r}(u). \tag{7.7}$$

where $A_{m,r} = NH_m\mathbf{F}_m cG_{rN}$, and $\xi_{m,r}(u) = \frac{1-e^{j2\pi miQ/P}}{1-e^{j2\pi(rN/Q+m/P)}}$. The mean square phase noise is then

$$\langle\phi^2[i]\rangle = [\sum_{u=0}^{iQ-1}\sum_{r=0}^{\lfloor\frac{Q-1}{N}\rfloor}\sum_{m=0}^{P-1}\mathbf{A}'_{m,r}\xi_{m,r}(u)]^2\boldsymbol{\sigma_c}^2 . \qquad (7.8)$$

where $\mathbf{A}'_{m,r} = NH_m\mathbf{F}_m G_{rN}$ and $\boldsymbol{\sigma_c}^2$ is the variance of Gaussian random variable c.

REFERENCES

[1] IEEE standard definitions of physical quantities for fundamentalfrequency and time metrology. Technical report, Instrumentation and Measurement Group of the IEEE Power Engineering Society (USA), 1989. Available at http://www.ieee.org/.

[2] K. Banerjee A. Koukab and M. Declercq. Modeling techniques and verification methodologies for substrate coupling effects in mixed-signal system-on-chip designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(6):823–836, June 2004.

[3] H. Aboushady, L. de Lamarre, N. Beilleau, and M. M. Louerat. Automatic synthesis and simulation of continuous-time sigma delta modulators. *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, 1:674–675, February 2004.

[4] Johnathan F. Adlard. *Frequency Shift Filtering for Cyclostationary Signals*. PhD thesis, University of York, 2000).

[5] Chirayu S. Amin, Masud H. Chowdhury, and Yehea I. Ismail. Realizable RLCK circuit crunching. In *DAC '03: Proceedings of the 40th conference on Design automation*, pages 226–231, New York, NY, USA, 2003. ACM Press.

[6] X. Aragones, J. l. Gonzalez, and A. Rubio. *Analysis and Solutions for Switching Noise Coupling in Mixed-Signal ICs*. Kluwer Academic Publishers, 1999.

[7] S. L. Avila, A. C. Lisboa, L. Krahenbuhl, W. P. Jr. Carpes, J. A. Vasconcelos, R. R. Saldanha, and R. H. C. Takahashi. Sensitivity analysis applied to decision making in multiobjective evolutionary optimization. *IEEE Transactions on Magnetics*, 42(4):1103–1106, April 2006.

[8] F. Balasa, S. C. Maruvada, and K. Krishnamoorthy. Efficient solution space exploration based on segment trees in analog placement with symmetry constraints.

2002. ICCAD 2002. IEEE/ACM International Conference on Computer Aided Design, pages 497–502, November 2002.

[9] F. Balasa, S.C. Maruvada, and K. Krishnamoorthy. Module placement for analog layout using the sequence-pair representation. In *IEEE International Conference on Computer Aided Design*, pages 497–502, November 2002.

[10] Roger Barth. ITRS commodity memory roadmap. In *MTDT '03: Proceedings of the 2003 International Workshop on Memory Technology, Design and Testing*, page 61, Washington, DC, USA, 2003. IEEE Computer Society.

[11] D. E. Best. *Phase-Locked Loop: Design, Simulation, and applications, (5th Ed.)*. McGraw-Hill, New York, 2003.

[12] Stephan Bourduas, Henry H. Y. Chan, and Zeljko Zilic. Blocking-aware task assignment for wormhole routed network-on-chip. In *IEEE Midwest Symposium on Circuits and Systems, 2007. MWSCAS 2007.*, Montreal, QC, may 2007.

[13] Stephen P. Boyd and Seung Jean Kim. Geometric programming for circuit optimization. In *ISPD '05: Proceedings of the 2005 international symposium on Physical design*, pages 44–46, New York, NY, USA, 2005. ACM Press.

[14] Ian Brynjolfson. Dynamic clock management circuits for low power applications. Master's thesis, McGill University, 2001.

[15] Cadence Design Systems. *Cadence Viruoso NeoCell.* Available at www.cadence.com.

[16] Henry H. Y. Chan. Substrate coupling analysis and noise reduction methods. Master's thesis, McGill University, 2001.

[17] Henry H. Y. Chan and Zeljko Zilic. A practical substrate modeling algorithm with active guardband macromodel for mixed-signal substrate coupling verification. In *ICECS 2001: The 8th IEEE International Conference on Electronics, Circuits and Systems*, volume 3, pages 1455–1460, Malta, 2001.

[18] Henry H. Y. Chan and Zeljko Zilic. Substrate coupled noise reduction and active noise suppressioncircuits for mixed-signal system-on-a-chip designs. In *MWSCAS 2001: Proceedings of the 44th IEEE 2001 Midwest Symposium on Circuits and Systems*, volume 1, pages 154–157, Dayton, OH, USA, 2001.

[19] Henry H. Y. Chan and Zeljko Zilic. Estimating phase-locked loop jitter due to substrate coupling: a cyclostationary approach. In *5th International Symposium on Quality Electronic Design, 2004. Proceedings.*, pages 309–314, 2004.

[20] Henry H. Y. Chan and Zeljko Zilic. IC parasitics estimation for analog circuit optimization. In *IRIS/PRECARN '04: Proceedings of IRIS/PRECARN 14th Annual. Canadian Conference on Intelligent Systems*, Ottawa, ON, Canada, 2004.

[21] Henry H. Y. Chan and Zeljko Zilic. Modeling layout effects for sensitivity-based analog circuit optimization. In *ISQED '05: Proceedings of the 6th International Symposium on Quality of Electronic Design*, pages 390–395, Washington, DC, USA, 2005. IEEE Computer Society.

[22] Henry H. Y. Chan and Zeljko Zilic. Modeling simultaneous switching noise-induced jitter for system-on-chip phase-locked loops. In *DAC '07: Proceedings of the 44th annual conference on Design automation*, pages 430–435, New York, NY, USA, 2007. ACM Press.

[23] Henry H. Y. Chan and Zeljko Zilic. Parasitic-aware physical design optimization of deep sub-micron analog circuits. In *IEEE Midwest Symposium on Circuits and Systems, 2007. MWSCAS 2007.*, Montreal, QC, may 2007.

[24] Henry H. Y. Chan and Zeljko Zilic. A performance driven layout compaction optimization algorithm for analog circuits. In *IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007.*, pages 2934–2937, New Orleans, LA, May 2007.

[25] Luca Daniel, Alberto Sangiovanni-Vincentelli, and Jacob White. Proximity templates for modeling of skin and proximity effects on packages and high frequency interconnect. In *ICCAD '02: Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, pages 326–333, New York, NY, USA, 2002. ACM Press.

[26] Maria del Mar Hershenson. Efficient description of the design space of analog circuits. In *DAC '03: Proceedings of the 40th conference on Design automation*, pages 970–973, New York, NY, USA, 2003. ACM Press.

[27] Mohamed Dessouky. *Design for Reuse of Analog Circuits. Case Study: Very Low-Voltage Delta-Sigma Modulator.* PhD thesis, University of Paris VI, 2001).

[28] R. Rutenbar L.R. Carley E. Ochotta, T. Mukherjee. *Practical Synthesis of High-Performance Analog Circuits*. Kluwer Academic Publishers, 1998.

[29] E. Razavi (Ed). *Monolithic Phase-Locked Loops and CLock Recovery Circuits: Theory and Design.* IEEE Press, 1996.

[30] J. Cong et al. Dynamic weighting Monte Carlo for constrained floorplan designs in mixed signal application. In *Proceedings of the ASP Design Automation Conference*, pages 277–282, January 2000.

[31] E. Habekotte, B. Hoefflinger, H. W. Klein, and M. A. Beunder. State of the art in the analog CMOS circuit design. *Proceedings of the IEEE*, 75(6):816–828, June 1987.

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

[33] F. Herzel and B. Razavi. A study of oscillator jitter due to supply and substrate noise. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on [see also Circuits and Systems II: Express Briefs, IEEE Transactions on]*, 46(1):56–62, January 1999.

[34] Hisamoto Hiyoshi and Kokichi Sugihara. A sequence of generalized coordinate systems based on Voronoi diagrams and its application to interpolation. In *GMP '00: Proceedings of the Geometric Modeling and Processing 2000*, page 129, Washington, DC, USA, 2000. IEEE Computer Society.

[35] H. Janardhan and M. F. Wagdy. Design of a 1GHz digital PLL using 0.18μm CMOS technology. In *Information Technology: New Generations, 2006. ITNG 2006. Third International Conference on*, pages 599–600, April 2006.

[36] N. J. Rohrer K. Bernstein. *SOI Circuit Design Concepts.* Kluwer Academic Publishers, 2000.

[37] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[38] A. Levy and M. Lindenbaum. Efficient sequential Karhunen-Loeve basis extraction and its application to images. *IEEE Transactions on Image Processing*, 9(8):1371–1374, August 2000.

[39] Xin Li, P. Gopalakrishnan, Yang Xu, and T. Pileggi. Robust analog/rf circuit design with projection-based posynomial modeling. *2004. ICCAD-2004. IEEE/ACM International Conference on Computer Aided Design*, pages 855–862, November 2004.

[40] C. Lin and D. M. W. Leenaerts. A new efficient method for substrate-aware device-level placement (short paper). In *ASP-DAC '00: Proceedings of the 2000 conference*

on Asia South Pacific design automation, pages 533–536, New York, NY, USA, 2000. ACM Press.

[41] L. Scheffer LLS. Lavagno, G. Martin. *Electronic Design Automation for Integrated Circuits Handbook.* Taylor and Francis group, CRC Press, 2006.

[42] M. M. Louerat, et al. Universit Pierre et Marie Curie. *The CAIRO+ Project : Creating Analog IPs - Reusable and Optimized.* Available at http://www-asim.lip6.fr/recherche/analog/cairo/.

[43] David Maliniak. It's time to get 'in the know' to tame nanometer effects. Technical report, Electronic Design (online), 2003. Available at http://www.elecdesign.com/Articles/Index.cfm?AD=1&ArticleID=5086.

[44] P. C. Maulik and L. R. Carley. High-performance analog module generation using non-linear optimization. In *ASIC Conference and Exhibit, 1991. Proceedings., Fourth Annual IEEE International*, pages 13–5, Rochester, NY, USA, September 1991.

[45] Trent McConaghy, Pieter Palmers, Georges Gielen, and Michiel Steyaert. Simultaneous multi-topology multi-objective sizing across thousands of analog circuit topologies. In *DAC '07: Proceedings of the 44th annual conference on Design automation*, pages 944–947, New York, NY, USA, 2007. ACM Press.

[46] R. Naiknaware and T.S. Fiez. Automated hierarchical CMOS analog circuit stack generation with intramodule connectivity and matching considerations. *IEEE Journal of Solid-State Circuits*, 34(3).

[47] S. Nazarian, M. Pedram, E. Tuncer, and T. Lin. Sensitivity-based gate delay propagation in static timing analysis. In *ISQED: Sixth International Symposium on Quality of Electronic Design*, pages 536–541, March 2005.

[48] D.A. Neamen. *Semiconductor Physics And Devices, (2/e).* Irwin/McGraw-Hill, 1997.

[49] E. S. Ochotta, R. A. Rutenbar, and L. R. Carley. Synthesis of high-performance analog circuits in ASTRX/OBLX. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(3):273–294, March 1996.

[50] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers. Matching properties of MOS transistors. *IEEE Journal of Solid-State Circuits*, 24(5):1433–1439, October 1989.

[51] R. Phelps, M. Krasnicki, R. A. Rutenbar, L. R. Carley, and J. R. Hellums. Anaconda: simulation-based synthesis of analog circuits via stochastic pattern search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(6):703–717, June 2000.

[52] T. Pialis and K. Phang. Analysis of timing jitter in ring oscillators due to power supply noise. In *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, volume 1, May 2003.

[53] Python Software Foundation. *Python Tutorial*. Available at www.python.org.

[54] Dave Reed. Keeping leakage current under control. Technical report, Electronic Design (online), 2003. Available at http://www.eetimes.com/story/OEG20030207S0026.

[55] R. A. Rutenbar. Design automation for analog: The next generation of tool challenges. In *Computer-Aided Design, 2006. ICCAD '06. IEEE/ACM International Conference on*, pages 458–460, San Jose, CA, November 2006.

[56] L. Vandenberghe S. Boyd, S. J. Kim and A. Hassibi. A tutorial on geometric programming. Technical report, Stanford University EE, USA, 2004.

[57] A. Samavedam and et. al. A scalable substrate noise coupling model for design of mixed-signal ic's. *IEEE Journal of Solid-State Circuits*, 35(6).

[58] Hanan Samet. Data structures for quadtree approximation and compression. *Commun. ACM*, 28(9):973–993, 1985.

[59] J. Sercu and S. Hammadi. Minimal-order multi-dimensional linear interpolation for a parameterized electromagnetic model database. *2003 IEEE MTT-S International Microwave Symposium Digest*, 1:295–298, June 2003.

[60] Erchin Serpedin, Flaviu Panduru, Ilkay Sari, and Georgios B. Giannakis. Bibliography on cyclostationarity. *Signal Process.*, 85(12):2233–2303, 2005.

[61] J.A.McCall S.H. Hall, G.W.Hall. *High-Speed Digital System Design – A handbook of interconnect theory and design practices*. John Wiley and Sons, Inc., 2000.

[62] N. A. Sherwani. *Algorithms for VLSI Physical Design Automation (3rd Ed.)*. Kluwer Academic Publishers, 1999.

[63] L. Miguel Silveira, Mattan Kamon, Ibrahim Elfadel, and Jacob White. A coordinate-transformed arnoldi algorithm for generating guaranteed stable reduced-order models of RLC circuits. In *ICCAD '96: Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*, pages 288–294, Washington, DC, USA, 1996. IEEE Computer Society.

[64] Synopsys Inc. *Synopsys Circuit Explorer*. Available at www.synopsys.com.

[65] M. Taherzadeh-Sani, R. Lotfi, H. Zare-Hoseini, and O. Shoaei. Design optimization of analog integrated circuits using simulation-based genetic algorithm. In *Signals, Circuits and Systems, 2003. SCS 2003. International Symposium on*, volume 1, pages 73–76, July 2003.

[66] Hua Tang, Hui Zhang, and Alex Doboli. Layout-aware analog system synthesis based on symbolic layout description and combined block parameter exploration, placement and global routing. In *ISVLSI '03: Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, page 266, Washington, DC, USA, 2003. IEEE Computer Society.

[67] Nishath K. Verghese, Timothy J. Schmerbeck, and David J. Allstot. *Simulation Techniques and Solutions for Mixed-Signal Coupling in Integrated Circuits*. Kluwer Academic Publishers, Norwell, MA, USA, 1995.

[68] Jiri Vlach and Kishore Singhal. *Computer Methods for Circuit Analysis and Design*. John Wiley & Sons, Inc., New York, NY, USA, 1983.

[69] Martin Vogels and Georges Gielen. Architectural selection of a/d converters. In *DAC '03: Proceedings of the 40th conference on Design automation*, pages 974–977, New York, NY, USA, 2003. ACM Press.

[70] Igor Vytyaz, David C. Lee, Suihua Lu, Amit Mehrotra, Un-Ku Moon, and Kartikeya Mayaram. Parameter finding methods for oscillators with a specified oscillation frequency. In *DAC '07: Proceedings of the 44th annual conference on Design automation*, pages 424–429, New York, NY, USA, 2007. ACM Press.

[71] M. Weiser. Hot topics-ubiquitous computing. *Computer*, 26(10):71–72, October 1993.

[72] Ke-Li Wu, Yong-Jiu Zhao, Jie Wang, and M. K. K. Cheng. An effective dynamic coarse model for optimization design of LTCC RF circuits with aggressive space

mapping. *IEEE Transactions on Microwave Theory and Techniques*, 52:393–402, January 2004.

[73] L. Zhang, R. Raut, Y. Jiang, and U. Kleine. Placement algorithm in analog-layout designs. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 25(10).