SECTION I

INTRODUCTION

1

Overview

Luciano Lavagno
Cadence Berkeley Laboratories
Berkeley, California

Grant Martin
Tensilica Inc.
Santa Clara, California

Louis Scheffer
Cadence Design Systems
San Jose, California

Introduction to Electronic Design Automation for Integrated Circuits

Modern integrated circuits (ICs) are enormously complicated, often containing many millions of devices. Design of these ICs would not be humanly possible without software (SW) assistance at every stage of the process. The tools used for this task are collectively called electronic design automation (EDA).

EDA tools span a very wide range, from purely logical tools that implement and verify functionality, to purely physical tools that create the manufacturing data and verify that the design can be manufactured. The next chapter, *The IC Design Process and EDA*, by Robert Damiano and Raul Camposano, discusses the IC design process, its major stages and design flow, and how EDA tools fit into these processes and flows. It particularly looks at interfaces between the major IC design stages and the kind of information — abstractions upwards, and detailed design and verification information downwards — that must flow between these stages.

A Brief History of Electronic Design Automation

This section contains a *very* brief summary of the origin and history of EDA for ICs. For each topic, the title of the relevant chapter(s) is mentioned in *italics.*

The need for tools became clear very soon after ICs were invented. Unlike a breadboard, ICs cannot be modified easily after fabrication, so testing even a simple change involves weeks of delay (for new masks and a new fabrication run) and considerable expense. Furthermore, the internal nodes of an IC are difficult to probe because they are physically small and may be covered by other layers of the IC. Even if these problems can be worked around, the internal nodes often have very high impedances and hence are difficult to measure without dramatically changing the performance. Therefore circuit simulators were crucial to IC design almost as soon as ICs came into existence. These programs are covered in the chapter *Analog Simulation: Circuit Level and Behavioral Level*, and appeared in the 1960s.

Next, as the circuits grew bigger, clerical help was required in producing the masks. At first there were digitizing programs, where the designer still drew with colored pencils but the coordinates were transferred to the computer, written to magnetic tape, and then transferred to the mask making machines. Soon, these early programs were enhanced into full-fledged layout editors. These programs were first developed in the late 1960s and early 1970s. Analog designs in the modern era are still largely laid out manually, with some tool assistance, as *Layout Tools for Analog ICs and Mixed-Signal SoCs: A Survey* will attest, although some developments in more automated optimization have been occurring, along with many experiments in more automated layout techniques.

As the circuits grew larger, getting the logic design correct became difficult, and *Digital Simulation* (i.e., logic simulation) was introduced into the IC design flow. Also, testing of the completed chip proved to be difficult, since unlike circuit boards, internal nodes could not be observed or controlled through a "bed of nails" fixture. Therefore automatic test pattern generation (ATPG) programs were developed that generate test vectors that only refer to the visible pins. Other programs that modified designs to make them more controllable, observable, and testable were not far behind. These programs, covered in *Design-for-Test* and *Automatic Test Pattern Generation*, were first available in the mid-1970s. Specialized *Analog and Mixed-Signal Test* needs were met by special testers and tools.

As the number of design rules, number of layers, and chip sizes all continued to increase, it became increasingly difficult to verify by hand that a layout met all the manufacturing rules, and to estimate the parasitics of the circuit. Therefore *Design Rule Checking*, and *Layout Extraction* programs were developed, starting in the mid-1970s. As the processes became more complex, with more layers of interconnect, the original analytic approximations to R, C, and L values became inadequate, and *High-Accuracy Parasitic Extraction* programs were required to determine more accurate values, or at least calibrate the parameter extractors.

The next bottleneck was doing the detailed designing of each polygon. Placement and routing programs allowed the user to specify only the gate-level netlist — the computer would then decide on the location of the gates and the wires connecting them. Although some silicon efficiency was lost, productivity was greatly improved, and IC design opened up to a wider audience of logic designers. The chapters *Digital Layout — Placement* and *Routing* cover these programs, which became popular in the mid-1980s.

Even just the gate-level netlist soon proved to be of too much detail, and synthesis tools were developed to create such a netlist from a higher level specification, usually expressed in a hardware description language (HDL). This is called *Logic Synthesis* and became available in the mid-1980s. In the last decade, *Power Analysis and Optimization from Circuit to Register Transfer Levels* has become a major area of concern and is becoming the number one optimization criterion for many designs, especially portable and battery powered ones.

Around this time, the large collection of tools that need to be used to complete a single design became a serious problem. Electronic design automation *Design Databases* were introduced to cope with this problem. In addition, *Design Flows* began to become more and more elaborate in order to hook tools together, as well as to develop and support both methodologies and use models for specific design groups, companies, and application areas.

In the late 1990s, as the circuits continued to shrink, noise became a serious problem. Programs that analyzed power and ground networks, cross-talk, and substrate noise in systematic ways became commercially available. The chapters *Design and Analysis of Power Supply Networks*, *Mixed-Signal Noise Coupling in System-on-Chip Design: Modeling, Analysis and Validation*, and *Noise Considerations in Digital ICs* cover these topics.

Gradually through the 1990s and early 2000s, chips and processes became sufficiently complex that the designs that optimize yield were no longer only a minimization of size. *Design for Manufacturability in the Nanometer Era*, otherwise known as "Design for Yield", became a field of its own. Also in this time frame, the size of the features on the chip became comparable to, or less than, the wavelength of the light used to create them. To compensate for this as much as possible, the masks were no longer a direct copy of what the designer intended. The creation of these more complex masks is covered in *Resolution Enhancement Techniques and Mask Data Preparation.*

On a parallel track, developing the process itself was also a difficult problem. *Process Simulation* tools were developed to predict the effects of changing various process parameters. The output from these programs, such as doping profiles, was useful to process engineers but too detailed for electrical analysis. Another suite of tools (see *Device Modeling — from Physics to Electrical Parameter Extraction*) that predict device performance from a physical description of devices was needed and developed. These models were particularly useful when developing a new process.

One of the areas that developed very early in the design of electronic systems, at least in part, but which is the least industrialized as a standard process, is that of system-level design. As the chapter on *Using Performance Metrics to Select Microprocessor Cores for IC Designs* points out, one of the first instruction set simulators appeared soon after the first digital computers did. However, until the present day, system-level design has consisted mainly of a varying collection of tricks, techniques, and *ad hoc* modeling tools.

The logic simulation and synthesis processes introduced in the 1970s and 1980s, respectively, are, as was discussed earlier, much more standardized. The front–end IC design flow would not have been possible to standardize without the introduction of standard HDLs. Out of a huge variety of HDLs introduced from the 1960s to the 1980s, Verilog and VHDL have become the major *Design and Verification Languages.* For a long time — till the mid to late 1990s, verification of digital design seemed stuck at standard digital simulation — although at least since the 1980s, a variety of *Hardware Acceleration and Emulation* solutions have been available to designers. However, advances in verification languages and the growth in design complexity have triggered interest in more advanced verification methods, and the last decade has seen considerable interest in *Using Transactional Level Models in a SoC Design Flow*, *Assertion-based Verification*, and *Formal Property Verification. Equivalence Checking* has been the formal technique most tightly integrated into design flows, since it allows designs to be compared before and after various optimizations and back-end-related modifications, such as scan insertion.

For many years, specific systems design domains have fostered their own application-specific *Tools and Methodologies for System-Level Design* — especially in the areas of algorithm design from the late 1980s through to this day. The late 1990s saw the emergence of and competition between a number of C/C++-based *System-Level Specification and Modeling Languages.* With the possibility of now incorporating the major functional units of a design (processors, memories, digital and mixed-signal HW blocks, peripheral interfaces, and complex hierarchical buses) all onto a single silicon substrate, the mid-1990s to the present day have also seen the rise of the System-on-chip (SoC). It is thus that the area of *SoC Block-Based Design and IP Assembly* has grown, in which the complexity possible with advanced semiconductor processes is ameliorated to some extent via reuse of blocks of design. Concomitant with the SoC approach has been the development, during the last decade, of *Performance Evaluation Methods for MPSoC Design*, development of embedded processors through specialized *Processor Modelling and Design Tools,* and gradual and still-forming links to *Embedded Software Modelling and Design.* The desire to raise HW design productivity to higher levels has spawned considerable interest in (*Parallelizing*) *High Level Synthesis* over the years. It is now seeing something of a resurgence driven by C/C++/SystemC as opposed to the first-generation high-level synthesis (HLS) tools driven by HDLs in the mid-1990s.

After the system level of design, architects need to descend one level of abstraction to the micro-architectural level. Here, a variety of tools allow one to look at the three main performance criteria: timing or delay (*Cycle-accurate System-Level Modeling and Performance Evaluation*), power (*Micro-Architectural Power Estimation and Optimization*), and physical *Design Planning.* Micro-architects need to make trade-offs between the timing, power, and cost/area attributes of complex ICs at this level.

The last several years have seen a considerable infilling of the design flow with a variety of complementary tools and methods. Formal verification of function is only possible if one is assured that the timing is correct, and by keeping a lid on the amount of dynamic simulation required, especially at the postsynthesis and postlayout gate levels, good *Static Timing Analysis* tools provide the assurance that timing constraints are being met. It is also an underpinning to timing optimization of circuits and for the design of newer mechanisms for manufacturing and yield. Standard cell-based placement and routing are not appropriate for *Structured Digital Design* of elements such as memories and register files, leading to specialized tools. As design groups began to rely on foundries and application specific integrated circuit (ASIC) vendors and as the IC design and manufacturing industry began to "de-verticalize", design libraries, covered in *Exploring Challenges of Libraries for Electronic Design*, became a domain for special design flows and tools. It ensured the availability of a variety of high performance and low power libraries for optimal design choices and allowed some portability of design across processes and foundries. *Tools for Chip-Package Codesign* began to link more closely the design of IOs on chip, the packages they fit into, and the boards on which they would be placed. For implementation "fabrics" such as field-programmable gate arrays (FPGAs), specialized *FPGA Synthesis and Physical Design Tools* are necessary to ensure good results. And a renewed emphasis on *Design Closure* allows a more holistic focus on the simultaneous optimization of design timing, power, cost, reliability, and yield in the design process.

Another area of growing but specialized interest in the analog design domain is the use of new and higher level modeling methods and languages, which are covered in *Simulation and Modeling for Analog and Mixed-Signal Integrated Circuits.*

A much more detailed overview of the history of EDA can be found in [1]. A historical survey of many of the important papers from the International Conference on Computer-Aided Design (ICCAD) can be found in [2].

Major Industry Conferences and Publications

The EDA community, formed in the early 1960s from tool developers working for major electronics design companies such as IBM, AT&T Bell Labs, Burroughs, Honeywell, and others, has long valued workshops, conferences, and symposia, in which practitioners, designers, and later, academic researchers, could exchange ideas and practically demonstrate the techniques. The Design Automation Conference (DAC) grew out of workshops, which started in the early 1960s, and although held in a number of U.S. locations, has in recent years tended to stay on the west coast of the United States or a bit inland. It is the largest combined EDA trade show and technical conference held annually anywhere in the world. In Europe, a number of country-specific conferences held sporadically through the 1980s, and two competing ones held in the early 1990s, led to the creation of the consolidated Design Automation and Test in Europe (DATE) conference, which started in the mid-1990s and has grown consistently in strength ever since. Finally, the Asia-South Pacific DAC (ASP-DAC) started in the mid to late 1990s and completes the trio of major EDA conferences spanning the most important electronics design communities in the world.

Complementing the larger trade show/technical conferences has been ICCAD, which for over 20 years has been held in San Jose, and has provided a more technical conference setting for the latest algorithmic advances in EDA to be presented, attracting several hundred attendees. Various domain areas of EDA knowledge have sparked a number of other workshops, symposia, and smaller conferences over the last 15 years, including the International Symposium on Physical Design (ISPD), International Symposium on Quality in Electronic Design (ISQED), Forum on Design Languages in Europe (FDL), HDL and Design and Verification conferences (HDLCon, DVCon), High-level Design, Verification and Test (HLDVT), International Conference on Hardware–Software Codesign and System Synthesis (CODES+ISSS), and many other gatherings. Of course, the area of Test has its own long-standing International Test Conference (ITC); similarly, there are specialized conferences for FPGA design (e.g., Forum on Programmable Logic [FPL]) and a variety of conferences focusing on the most advanced IC

designs such as the International Solid–State Circuits Conference (ISSCC) and its European counterpart (ESSCC).

There are several technical societies with strong representation of design automation: one is the Institute of Electrical and Electronics Engineers (IEEE, pronounced "eye-triple-ee"), and the other is the Association for Computing Machinery (ACM).

Various IEEE and ACM transactions contain major work on algorithms and design techniques in print — a more archival-oriented format than conference proceedings. Among these, the IEEE Transactions on computer-aided design (CAD), the IEEE Transactions on VLSI systems, and the ACM Transactions on Design Automation of Electronic Systems are notable. A more general readership magazine devoted to Design and Test and EDA topics is IEEE Design and Test.

As might be expected, the EDA community has a strong online presence. All the conferences mentioned above have web pages describing locations, dates, manuscript submission and registration procedures, and often detailed descriptions of previous conferences. The journals above offer online submission, refereeing, and publication. Online, the IEEE (http://ieee.org), ACM (http://acm.org), and *CiteSeer* (http://citeseer.ist.psu.edu) offer extensive digital libraries, which allow searches through titles, abstracts, and full text. Both conference proceedings and journals are available. Most of the references found in this volume, at least those published after 1988, can be found in at least one of these libraries.

Structure of the Book

In the simplest case of digital design, EDA can be divided into system-level design, micro-architecture design, logical verification, test, synthesis-place-and-route, and physical verification. System-level design is the task of determining which components (bought and built, HW and SW) should comprise a system that can do what one wants. Micro-architecture design fills out the descriptions of each of the blocks, and sets the main parameters for their implementation. Logical verification verifies that the design does what is intended. Test ensures that functional and nonfunctional chips can be told apart reliably, and inserts testing circuitry if needed to ensure that this is the case. Synthesis, place, and route take the logical description, and map it into increasingly detailed physical descriptions, until the design is in a form that can be built with a given process. Physical verification checks that the design is manufacturable and will be reliable. In general, each of these stages works with an increasingly detailed description of the design, and may fail due to problems unforeseen at earlier stages. This makes the flow, or sequence of steps that the users follow to finish their design, a crucial part of any EDA methodology.

Of course not all, or even most chips, are fully digital. Analog chips and chips with a mixture of analog and digital signals (commonly called mixed-signal chips) require their own specialized tool sets.

All these tools must work on circuits and designs that are quite large, and do so in a reasonable amount of time. In general, this cannot be done without models, or simplified descriptions of the behavior of various chip elements. Creating these models is the province of Technology CAD (TCAD), which in general treats relatively small problems in great physical detail, starting from very basic physics and building the more efficient models needed by the tools that must handle higher data volumes.

The division of EDA into these sections is somewhat arbitrary, and below a brief description of each of the chapters of the book is given.

System Level Design

Tools and Methodologies for System-Level Design by Bhattacharyya and Wolf

This chapter covers very high level system-level design approaches and associated tools such as Ptolemy, the Mathworks tools, and many others, and uses video applications as a specific example illustrating how these can be used.

System-Level Specification and Modeling Languages by Buck

This chapter discusses the major approaches to specify and model systems, and the languages and tools in this domain. It includes issues of heterogeneous specifications, models of computation and linking multidomain models, requirements on languages, and specialized tools and flows in this area.

SoC Block-Based Design and IP Assembly by Wilson

This chapter approaches system design with particular emphasis on SoC design via IP-based reuse and block-based design. Methods of assembly and compositional design of systems are covered. Issues of IP reuse as they are reflected in system-level design tools are also discussed.

Performance Evaluation Methods for Multiprocessor Systems-on-Chip Design by Bacivarov and Jerraya

This chapter surveys the broad field of performance evaluation and sets it in the context of multi-processor systems-on-chip (MPSoC). Techniques for various types of blocks — HW, CPU, SW, and interconnect — are included. A taxonomy of performance evaluation approaches is used to assess various tools and methodologies.

System-Level Power Management by Chang, Macii, Poncino and Tiwari

This chapter discusses dynamic power management approaches, aimed at selectively stopping or slowing down resources, whenever this is possible while still achieving the required level of system performance. The techniques can be applied both to reduce power consumption, which has an impact on power dissipation and power supply, and energy consumption, which improves battery life. They are generally driven by the software layer, since it has the most precise picture about both the required quality of service and the global state of the system.

Processor Modeling and Design Tools by Mishra and Dutt

This chapter covers state-of-the-art specification languages, tools, and methodologies for processor development used in academia and industry. It includes specialized architecture description languages and the tools that use them, with a number of examples.

Embedded Software Modeling and Design by di Natale

This chapter covers models and tools for embedded SW, including the relevant models of computation. Practical approaches with languages such as unified modeling language (UML) and specification and description language (SDL) are introduced and how these might link into design flows is discussed.

Using Performance Metrics to Select Microprocessor Cores for IC Designs by Leibson

This chapter discusses the use of standard benchmarks, and instruction set simulators, to evaluate processor cores. These might be useful in nonembedded applications, but are especially relevant to the design of embedded SoC devices where the processor cores may not yet be available in HW, or be based on user-specified processor configuration and extension. Benchmarks drawn from relevant application domains have become essential to core evaluation and their advantages greatly exceed that of the general-purpose ones used in the past.

Parallelizing High-Level Synthesis: A Code Transformational Approach to High-Level Synthesis by Singh, Gupta, Shukla, and Gupta

This chapter surveys a number of approaches, algorithms, and tools for HLS from algorithmic or behavioral descriptions, and focuses on some of the most recent developments in HLS. These include the use of techniques drawn from the parallel compiler community.

Micro-Architecture Design

Cycle-Accurate System-Level Modeling and Performance Evaluation by Coppola and Grammatikakis

This chapter discusses how to use system-level modeling approaches at the cycle-accurate micro-architectural level to do final design architecture iterations and ensure conformance to timing and performance specifications.

Micro-Architectural Power Estimation and Optimization by Macii, Mehra, and Poncino

This chapter discusses the state of the art in estimating power at the micro-architectural level, consisting of major design blocks such as data paths, memories, and interconnect. *Ad hoc* solutions for optimizing both specific components and the whole design are surveyed.

Design Planning by Otten

This chapter discusses the topics of physical floor planning and its evolution over the years, from dealing with rectangular blocks in slicing structures to more general mathematical techniques for optimizing physical layout while meeting a variety of criteria, especially timing and other constraints.

Logical Verification

Design and Verification Languages by Edwards

This chapter discusses the two main HDLs in use — VHDL and Verilog, and how they meet the requirements for design and verification flows. More recent evolutions in languages, such as SystemC, System Verilog, and verification languages such as OpenVera, e, and PSL are also described.

Digital Simulation by Sanguinetti

This chapter discusses logic simulation algorithms and tools, as these are still the primary tools used to verify the logical or functional correctness of a design.

Using Transactional Level Models in a SoC Design Flow by Clouard, Ghenassia, Maillet-Contoz, and Strassen

This chapter discusses a real design flow at a real IC design company to illustrate the building, deployment, and use of transactional-level models to simulate systems at a higher level of abstraction, with much greater performance than at register transfer level (RTL), and to verify functional correctness and validate system performance characteristics.

Assertion-Based Verification by Foster and Marschner

This chapter introduces the relatively new topic of assertion-based verification, which is useful for capturing design intent and reusing it in both dynamic and static verification methods. Assertion libraries such as OVL and languages such as PSL and System Verilog assertions are used for illustrating the concepts.

Hardware Acceleration and Emulation by Bershteyn and Turner

This chapter discusses HW-based systems including FPGA, processor based accelerators/emulators, and FPGA prototypes for accelerated verification. It compares the characteristics of each type of system and typical use models.

Formal Property Verification by Fix and McMillan

This chapter discusses the concepts and theory behind formal property checking, including an overview of property specification and a discussion of formal verification technologies and engines.

Test

Design-for-Test by Koenemann

This chapter discusses the wide variety of methods, techniques, and tools available to solve design-for-test (DFT) problems. This is a huge area with a huge variety of techniques, many of which are implemented in tools that dovetail with the capabilities of the physical test equipment. The chapter surveys the specialized techniques required for effective DFT with special blocks such as memories as well as general logic cores.

Automatic Test Pattern Generation by Wang and Cheng

This chapter starts with the fundamentals of fault modeling and combinational ATPG concepts. It moves on to gate-level sequential ATPG, and discusses satisfiability (SAT) methods for circuits. Moving on beyond traditional fault modeling, it covers ATPG for cross talk faults, power supply noise, and applications beyond manufacturing test.

Analog and Mixed-Signal Test by Kaminska

This chapter first overviews the concepts behind analog testing, which include many characteristics of circuits that must be examined. The nature of analog faults is discussed and a variety of analog test equipment and measurement techniques surveyed. The concepts behind analog built-in-self-test (BIST) are reviewed and compared with the digital test.

RTL to GDS-II, or Synthesis, Place, and Route

Design Flows by Hathaway, Stok, Chinnery, and Keutzer

The RTL to GDSII flow has evolved considerably over the years, from point tools hooked loosely together, to a more integrated set of tools for design closure. This chapter addresses the design flow challenges based on the rising interconnect delays and new challenges to achieve closure.

Logic Synthesis by Khatri and Shenoy

This chapter provides an overview and survey of logic synthesis, which has since the early 1980s, grown to be the vital center of the RTL to GDSII design flow for digital design.

Power Analysis and Optimization from Circuit to Register Transfer Levels by Monteiro, Patel, and Tiwari

Power has become one of the major challenges in modern IC design. This chapter provides an overview of the most significant CAD techniques for low power, at several levels of abstraction.

Equivalence Checking by Kuehlmann and Somenzi

Equivalence checking can formally verify whether two design specifications are functionally equivalent. The chapter defines the equivalence-checking problem, discusses the foundation for the technology, and then discusses the algorithms for combinational and sequential equivalence checking.

Digital Layout — Placement by Reda and Kahng

Placement is one of the fundamental problems in automating digital IC layout. This chapter reviews the history of placement algorithms, the criteria used to evaluate quality of results, many of the detailed algorithms and approaches, and recent advances in the field.

Static Timing Analysis by Sapatnekar

This chapter overviews the most prominent techniques for static timing analysis. It then outlines issues relating to statistical timing analysis, which is becoming increasingly important to handle process variations in advanced IC technologies.

Structured Digital Design by Mo and Brayton

This chapter covers the techniques for designing regular structures, including data paths, programable logic arrays, and memories. It extends the discussion to include regular chip architectures such as gate arrays and structured ASICs.

Routing by Scheffer

Routing continues from automatic placement as a key step in IC design. Routing creates all the wires necessary to connect all the placed components while obeying the process design rules. This chapter discusses various types of routers and the key algorithms.

Exploring Challenges of Libraries for Electronic Design by Hogan and Becker

This chapter discusses the factors that are most important and relevant for the design of libraries and IP, including standard cell libraries, cores, both hard and soft, and the design and user requirements for the same. It also places these factors in the overall design chain context.

Design Closure by Cohn and Osler

This chapter describes the common constraints in VLSI design, and how they are enforced through the steps of a design flow that emphasizes design closure. A reference flow for ASIC is used and illustrated. Future design closure issues are also discussed.

Tools for Chip-Package Codesign by Franzon

Chip-package co-design refers to design scenarios, in which the design of the chip impacts the package design or vice versa. This chapter discusses the drivers for new tools, the major issues, including mixed-signal needs, and the major design and modeling approaches.

Design Databases by Bales

The design database is at the core of any EDA system. While it is possible to build a bad EDA tool or flow on *any* database, it is impossible to build a good EDA tool or flow on a bad database. This chapter describes the place of a design database in an integrated design system. It discusses databases used in the past, those currently in use as well as emerging future databases.

FPGA Synthesis and Physical Design by Betz and Hutton

Programmable logic devices, both complex programmable logic devices (CPLDs) and FPGAs, have evolved from implementing small glue-logic designs to large complete systems. The increased use of such devices — they now are the majority of design starts — has resulted in significant research in CAD algorithms and tools targeting programmable logic. This chapter gives an overview of relevant architectures, CAD flows, and research.

Analog and Mixed-Signal Design

Analog Simulation: Circuit Level and Behavioral Level by Mantooth and Roychowdhury

Circuit simulation has always been a crucial component of analog system design and is becoming even more so today. In this chapter, we provide a quick tour of modern circuit simulation. This includes starting on the ground floor with circuit equations, device models, circuit analysis, more advanced analysis techniques motivated by RF circuits, new advances in circuit simulation using multitime techniques, and statistical noise analysis.

Simulation and Modeling for Analog and Mixed-Signal Integrated Circuits by Gielen and Philips

This chapter provides an overview of the modeling and simulation methods that are needed to design and embed analog and RF blocks in mixed-signal integrated systems (ASICs, SoCs, and SiPs). The role of behavioral models and mixed-signal methods involving models at multiple hierarchical levels is covered. The generation of performance models for analog circuit synthesis is also discussed.

Layout Tools for Analog ICs and Mixed-Signal SoCs: A Survey by Rutenbar and Cohn

Layout for analog circuits has historically been a time-consuming, manual, trial-and-error task. In this chapter, we cover the basic problems faced by those who need to create analog and mixed-signal layout, and survey the evolution of design tools and geometric/electrical optimization algorithms that have been directed at these problems.

Physical Verification

Design Rule Checking by Todd, Grodd, and Fetty

After the physical mask layout is created for a circuit for a specific design process, the layout is measured by a set of geometric constraints or rules for that process. The main objective of design rule checking is to achieve high overall yield and reliability. This chapter gives an overview of design rule checking (DRC) concepts and then discusses the basic verification algorithms and approaches.

Resolution Enhancement Techniques and Mask Data Preparation by Schellenberg

With more advanced IC fabrication processes, new physical effects, which could be ignored in the past, are being found to have a strong impact on the formation of features on the actual silicon wafer. It is now essential to transform the final layout via new tools in order to allow the manufacturing equipment to deliver the new devices with sufficient yield and reliability to be cost-effective. This chapter discusses the compensation schemes and mask data conversion technologies now available to accomplish the new design for manufacturability (DFM) goals.

Design for Manufacturability in the Nanometer Era by Dragone, Guardiani, and Strojwas

Achieving high yielding designs in state-of-the-art IC process technology has become an extremely challenging task. Design for manufacturability includes many techniques to modify the design of ICs in order to improve functional and parametric yield and reliability. This chapter discusses yield loss mechanisms and fundamental yield modeling approaches. It then discusses techniques for functional yield maximization and parametric yield optimization. Finally, DFM-aware design flows and the outlook for future DFM techniques are discussed.

Design and Analysis of Power Supply Networks by Blaauw, Pant, Chaudhry, and Panda

This chapter covers design methods, algorithms, tools for designing on-chip power grids, and networks. It includes the analysis and optimization of effects such as voltage drop and electro-migration.

Noise Considerations in Digital ICs by Kariat

On-chip noise issues and impact on signal integrity and reliability are becoming a major source of problems for deep submicron ICs. Thus the methods and tools for analyzing and coping with them, which are discussed in this chapter, have been gaining importance in recent years.

Layout Extraction by Kao, Lo, Basel, Singh, Spink, and Scheffer

Layout extraction is the translation of the topological layout back into the electrical circuit it is intended to represent. This chapter discusses the distinction between designed and parasitic devices, and discusses the three major parts of extraction: designed device extraction, interconnect extraction, and parasitic device extraction.

Mixed-Signal Noise Coupling in System-on-Chip Design: Modeling, Analysis, and Validation by Vergese and Nagata

This chapter describes the impact of noise coupling in mixed-signal ICs, and reviews techniques to model, analyze, and validate it. Different modeling approaches and computer simulation methods are presented, along with measurement techniques. Finally, the chapter reviews the application of substrate noise analysis to placement and power distribution synthesis.

Technology Computer-Aided Design

Process Simulation by Johnson

Process simulation is the modeling of the fabrication of semiconductor devices such as transistors. The ultimate goal is an accurate prediction of the active dopant distribution, the stress distribution,

and the device geometry. This chapter discusses the history, requirements, and development of process simulators.

Device Modeling — from Physics to Electrical Parameter Extraction by Dutton, Choi, and Kan

Technology files and design rules are essential building blocks of the IC design process. Development of these files and rules involves an iterative process that crosses the boundaries of technology and device development, product design, and quality assurance. This chapter starts with the physical description of IC devices and describes the evolution of TCAD tools.

High-Accuracy Parasitic Extraction by Kamon and Iverson

This chapter describes high-accuracy parasitic extraction methods using fast integral equation and random walk-based approaches.

References

[1] A. Sangiovanni-Vincentelli, The tides of EDA, *IEEE Des. Test Comput.*, 20, 59–75, 2003.
[2] A. Kuelhmann, Ed., *20 Years of ICCAD*, Kluwer Academic Publishers (now Springer), Dordrecht, 2002.