Overview

1

Luciano Lavagno, Grant E. Martin, Louis K. Scheffer, and Igor L. Markov

CONTENTS

1.1 INTRODUCTION TO *ELECTRONIC DESIGN AUTOMATION FOR INTEGRATED CIRCUITS HANDBOOK,* SECOND EDITION

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

EDA tools span a very wide range, from logic-centric tools that implement and verify functionality to physically-aware tools that create blueprints for manufacturing and verify their feasibility. EDA methodologies combine multiple tools into EDA design flows, invoking the most appropriate software packages based on how the design progresses through optimizations. Modern EDA methodologies can reuse existing design blocks, develop new ones, and integrate entire systems. They not only automate the work of circuit engineers, but also process large amounts of heterogeneous design data, invoke more accurate analyses and more powerful optimizations than what human designers are capable of.

1.1.1 BRIEF HISTORY OF ELECTRONIC DESIGN AUTOMATION

The need for design tools became clear soon after ICs were invented. Unlike a breadboard, an IC cannot be modified easily after fabrication; therefore, testing even a simple change takes weeks (for new masks and a new fabrication run) and requires considerable expense. 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. Internal nodes with high impedances are difficult to measure without dramatically changing the performance. Therefore, circuit simulators became crucial to IC design almost as soon as ICs came into existence. These programs appeared in the 1960s and are covered in Chapter 17 of *Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology* (hereafter referred to as Volume 2 of this Handbook).

As the circuits grew larger, clerical help was required in producing the masks. At first, the designer drew shapes with colored pencils but the coordinates were transferred to the computer by digitizing programs, written to magnetic tape (hence the handoff from design to fabrication is still called "tapeout"), and then transferred to the mask-making machines. In the 1960s and 1970s, these early programs were enhanced to full-fledged layout editors. Analog designs in the modern era are still largely laid out manually, with some tool assistance, as Chapter 19 of Volume 2 will attest.

As the circuits scaled up further, ensuring the correctness of logic designs became difficult, and logic simulation (Chapter 16) was introduced into the IC design flow. Testing completed chips proved difficult too, 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 to generate test vectors that can be entered through accessible pins. Other techniques that modified designs to make them more controllable, observable, and testable were not far behind. These techniques, covered in Chapters 21 and 22, were first available in the mid-1970s. Specialized needs were met by special testers and tools, discussed in Chapter 23.

As the number of design rules, number of layers, and chip size 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, as demonstrated in Chapters 20 and 25 of Volume 2 new software was developed, starting in the mid-1970s, to address this need. Increasing numbers of interconnect layers made the process more complex, and the original analytic approximations to R, C, and L values became inadequate, and new techniques for parasitic extraction were required to determine more accurate values, or at least calibrate the parameter extractors.

The next bottleneck was in determining the precise location of each polygon and drawing its detailed geometry. Placement and routing programs for standard-cell designs allowed the user to specify only the gate-level netlist—the computer would then decide on the location of the gates and route the wires connecting them. This greatly improved productivity (with a moderate loss of silicon efficiency), making IC design accessible to a wider group of electronics engineers. Chapters 5 and 8 of Volume 2 cover these programs, which became popular in the mid-1980s.

Even just the gate-level netlist soon proved unwieldy, and synthesis tools were developed to create such a netlist from a higher-level specification, usually expressed in a hardware description language (HDL). This step is called *Logic Synthesis*. It became available in the mid-1980s. In the late 2000s, the issues described in Chapter 2 of Volume 2 have become a major area of concern and the main optimization criterion, respectively, for many designs, especially in the portable and battery-powered categories. Around this time, the large collections of disparate tools required to complete a single design became a serious problem. Electronic design automation *Design Databases* were introduced as common infrastructure for developing interoperable tools. In addition, the techniques described in Chapter 1 of Volume 2 grew more elaborate in how tools were linked together to support design methodologies, as well as use models for specific design groups, companies, and application areas.

In the late 1990s, as transistors continued to shrink, electromagnetic noise became a serious problem. Programs that analyzed power and ground networks, cross-talk, and substrate noise in systematic ways became commercially available. Chapters 23, 24, and 26 of Volume 2 cover these topics.

Gradually through the 1990s and early 2000s, chips and design processes became so complex that yield optimization developed into a separate field called *Design for Manufacturability in the Nanometer Era*, otherwise known as "Design for Yield." In this time frame, the smallest on-chip features dropped below the wavelength of the light used to manufacture them with optical lithography. Due to the diffraction limit, the masks could no longer faithfully copy what the designer intended. The creation of these more complex masks is covered under Chapter 21 of Volume 2.

Developing the manufacturing process itself was also challenging. *Process Simulation* (Volume 2, Chapter 27) tools were developed to explore sensitivities to process parameters. The output of these programs, such as doping profiles, was useful to process engineers but too detailed for electrical analysis. A newly developed suite of tools (see Volume 2, Chapter 28) predicted device performance from a physical description of devices. These models were particularly useful when developing a new process.

System-level design became useful very early in the history of design automation. However, due to the diversity of application-dependent issues that it must address, it is also the least standardized level of abstraction. As Chapter 10 points out, one of the first instruction set simulators appeared soon after the first digital computers did. Yet, 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 than system-level design. The front-end IC design flow would have been much more difficult without 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 (Chapter 15). Until the late 1990s, verification of digital designs seemed stuck at standard digital simulation—although at least since the 1980s, a variety of Hardware-Assisted Verification and Software Development (Chapter 19) solutions have been available to designers. However, advances in verification languages and growing design complexity have motivated more advanced verification methods, and the last decade has seen considerable interest in Leveraging Transactional-Level Models in a SoC Design Flow (Chapter 17), Assertion-Based Verification (Chapter 18), and Formal Property Verification (Chapter 20). Equivalence Checking (Volume 2, Chapter 4) 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 system-design domains have fostered their own application-specific Tools and Methodologies for System-Level Design (Chapter 3)—especially in the areas of algorithm design from the late 1980s 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 (Chapter 4). With the newly available possibility to incorporate all major functional units of a design (processors, memories, digital and mixed-signal HW blocks, peripheral interfaces, and complex hierarchical buses) onto a single silicon substrate, the last 20 years have seen the rise of the system on chip (SoC). Thus, the area of SoC Block-Based Design and IP Assembly (Chapter 5) has grown, enabling greater complexity with advanced semiconductor processes through the reuse of design blocks. Along with the SoC approach, the last decade saw the emergence of Performance Evaluation Methods for MPSoC Designs (Chapter 6), development of embedded processors through specialized Processor Modeling and Design Tools (Chapter 8), and gradual and still-forming links to Models and Tools for Complex Embedded Software and Systems (Chapter 9*)*. The desire to improve HW design productivity has spawned considerable interest in High-Level Synthesis (Chapter 11) over the years. It is now experiencing 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 by one level of abstraction to the microarchitectural level. Here, a variety of tools allow one to look at the three main criteria: timing or delay (Microarchitectural and System-Level Power Estimation and Optimization), power (Chapter 13), and area and cost (Chapter 14). Microarchitects 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 variety of complementary tools and methods added to conventional design flows. Formal verification of design function is only possible if correct timing is guaranteed, and by limiting the amount of dynamic simulation required, especially at the postsynthesis and postlayout gate levels, Static Timing Analysis (Volume 2, Chapter 6) tools provide the assurance that timing constraints are met. Timing analysis also underlies timing optimization of circuits and the design of newer mechanisms for manufacturing and yield. Standard cell–based placement and routing are not appropriate for Structured Digital Design (Volume 2, Chapter 7) of elements such as memories and register files, and this observation motivates specialized tools. As design groups began to rely on foundries and application-specific (ASIC) vendors and as the IC design and manufacturing industry began to "deverticalize," design libraries, covered in Chapter 12 of Volume 2, became a domain for special design flows and tools. Library vendors offered 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 Co-Design (Volume 2, Chapter 14) 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 (Volume 2, Chapter 16) tools are necessary to ensure good results. A renewed emphasis on Design Closure (Volume 2, Chapter 13) 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 Chapter 18 of Volume 2.

Since the first edition of this handbook appeared, several new areas of design have reached a significant level of maturity. Gate Sizing (Volume 2, Chapter 10) techniques choose the best widths for transistors in order to optimize performance, both in a continuous setting (full-custom-like) and in a discrete setting (library based and FinFET based). Clock Design and Synthesis (Volume 2, Chapter 11) techniques enable the distribution of reliable synchronization to huge numbers of sequential elements. Finally, three-dimensional (3D) integrated circuits are attempting to extend the duration of Moore's law, especially in the elusive domain of improving performance, by allowing multiple ICs to be stacked on top of each other.

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

1.1.2 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, and Honeywell. It 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 US 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 conference, which started in the mid-1990s and has grown consistently in strength ever since. Finally, the Asia-South Pacific DAC started in the mid-1990s to late 1990s and completes the trio of major EDA conferences spanning the most important electronics design communities in the world.

Large trade shows and technical conferences have been complemented by ICCAD, held in San Jose for over 20 years. It has provided a more technical conference setting for the latest algorithmic advances, attracting several hundred attendees. Various domain areas of EDA knowledge have sparked a number of other workshops, symposia, and smaller conferences over the last 20 years, including the International Symposium on Physical Design, 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 and its European counterpart the European Solid-State Circuits Conference.

There are several technical societies with strong representation of design automation: one is the Institute of Electrical and Electronics Engineers (IEEE, pronounced as "eye-triple-ee") and the other is the Association for Computing Machinery (ACM). The Electronic Design Automation Consortium (EDAC) is an industry group that cosponsors major conferences such as DAC with professional societies.

Various IEEE and ACM transactions publish research on algorithms and design techniques—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 less-technical, broader-interest magazine is *IEEE Design and Test*.

As might be expected, the EDA community has a strong online presence. All the conferences have Web pages describing locations, dates, manuscript submission and registration procedures, and often detailed descriptions of previous conferences. The journals 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 texts. 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.

1.1.3 STRUCTURE OF THE BOOK

In the next chapter, "Integrated Circuit Design Process and Electronic Design Automation," Damiano, Camposano, and Martin discuss the IC design process, its major stages and design flow, and how EDA tools fit into these processes and flows. It particularly covers interfaces between the major IC design stages based on higher-level abstractions, as well as detailed design and verification information. Chapter 2 concludes the introductory section to the Handbook. Beyond that, *Electronic Design Automation for Integrated Circuits Handbook*, Second Edition, comprises several sections and two volumes. Volume 1 (in your hands) is entitled *Electronic Design Automation for IC System Design, Verification, and Testing*). Volume 2 is *Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology*. We will now discuss the division of these two books into sections.

EDA for digital design can be divided into system-level design, microarchitecture design, logic 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 perform required functions. Microarchitecture design fills out the descriptions of each of the blocks and sets the main parameters for their implementation. Logic verification checks that the design does what is intended. Postfabrication test ensures that functional and nonfunctional chips can be distinguished reliably. It is common to insert dedicated circuitry to make test efficient. Synthesis, placement, and routing take the logical design 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 such a design is manufacturable and will be reliable. This makes the design flow, or sequence of steps that the users follow to finish their design, a crucial part of any EDA methodology.

In addition to fully digital chips, analog and mixed-signal chips require their own specialized tool sets.

All these tools must scale to large designs and do so in a reasonable amount of time. In general, such scaling cannot be accomplished without behavioral models, that is, 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 problem instances 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. In the following, we give a brief description of each book chapter.

1.2 SYSTEM-LEVEL DESIGN

1.2.1 TOOLS AND METHODOLOGIES FOR SYSTEM-LEVEL DESIGN

Chapter 3 by Bhattacharyya and Wolf covers system-level design approaches and associated tools such as Ptolemy and the MathWorks tools, and illustrates them for video applications.

1.2.2 SYSTEM-LEVEL SPECIFICATION AND MODELING LANGUAGES

Chapter 4 by Edwards and Buck discusses major approaches to specifying and modeling systems, as well as the languages and tools in this domain. It covers heterogeneous specifications, models of computation and linking multidomain models, requirements on languages, and specialized tools and flows in this area.

1.2.3 SoC BLOCK-BASED DESIGN AND IP ASSEMBLY

Chapter 5 by Kashai approaches system design with particular emphasis on SoCs 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.

1.2.4 PERFORMANCE EVALUATION METHODS FOR MULTIPROCESSOR SYSTEMS-ON-CHIP DESIGN

Chapter 6 by Jerraya and Bacivarov surveys the broad field of performance evaluation and sets it in the context of multiprocessor system 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.

1.2.5 SYSTEM-LEVEL POWER MANAGEMENT

Chapter 7 by Chang, Macii, Poncino, and Tiwari discusses dynamic power management approaches, aimed at selectively stopping or slowing down resources, whenever possible while providing required levels of system performance. The techniques can be applied to reduce both power consumption and energy consumption, which improves battery life. They are generally driven by the SW layer, since it has the most precise picture about both the required quality of service and the global state of the system.

1.2.6 PROCESSOR MODELING AND DESIGN TOOLS

Chapter 8 by Chattopadhyay, Dutt, Leupers, and Mishra 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.

1.2.7 MODELS AND TOOLS FOR COMPLEX EMBEDDED SOFTWARE AND SYSTEMS

Chapter 9 by Di Natale covers models and tools for embedded SW, including the relevant models of computation. Practical approaches with languages such as Simulink® and the Unified Modeling Language are introduced. Embeddings into design flows are discussed.

1.2.8 USING PERFORMANCE METRICS TO SELECT MICROPROCESSOR CORES FOR IC DESIGNS

Chapter 10 by Leibson 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 configurations and extensions.

Benchmarks drawn from relevant application domains have become essential to core evaluation, and their advantages greatly exceed those of the general-purpose benchmarks used in the past.

1.2.9 HIGH-LEVEL SYNTHESIS

Chapter 11 by Balarin, Kondratyev, and Watanabe describes the main steps taken by a HLS tool to synthesize a C/C++/SystemC model into register transfer level (RTL). Both algorithmic techniques and user-level decisions are surveyed.

1.3 MICROARCHITECTURE DESIGN

1.3.1 BACK-ANNOTATING SYSTEM-LEVEL MODELS

Chapter 12 by Grammatikakis, Papagrigoriou, Petrakis, and Coppola discusses how to use system-level modeling approaches at the cycle-accurate microarchitectural level to perform final design architecture iterations and ensure conformance to timing and performance specifications.

1.3.2 MICROARCHITECTURAL POWER ESTIMATION AND OPTIMIZATION

Chapter 13 by Macii, Mehra, Poncino, and Dick discusses power estimation at the microarchitectural level in terms of data paths, memories, and interconnect. *Ad hoc* solutions for optimizing both specific components and entire designs are surveyed, with a particular emphasis on SoCs for mobile applications.

1.3.3 DESIGN PLANNING AND CLOSURE

Chapter 14 by Otten 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.

1.4 LOGIC VERIFICATION

1.4.1 DESIGN AND VERIFICATION LANGUAGES

Chapter 15 by Edwards 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, SystemVerilog, and verification languages (i.e., OpenVera, e, and PSL), are also described.

1.4.2 DIGITAL SIMULATION

Chapter16 by Sanguinetti discusses logic simulation algorithms and tools, as these are still the primary tools used to verify the logical or functional correctness of a design.

1.4.3 LEVERAGING TRANSACTIONAL-LEVEL MODELS IN A SoC DESIGN FLOW

In Chapter 17, Maillet-Contoz, Cornet, Clouard, Paire, Perrin, and Strassen focus on an industry design flow at a major IC design company to illustrate the construction, deployment, and use of transactional-level models to simulate systems at a higher level of abstraction, with much greater performance than at RTL, and to verify functional correctness and validate system performance characteristics.

1.4.4 ASSERTION-BASED VERIFICATION

Chapter 18 by Foster and Marschner introduces the 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 including PSL and SystemVerilog assertions are used for illustrating the concepts.

1.4.5 HARDWARE-ASSISTED VERIFICATION AND SOFTWARE DEVELOPMENT

Chapter 19 by Schirrmeister, Bershteyn, and Turner 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.

1.4.6 FORMAL PROPERTY VERIFICATION

In Chapter 20, Fix, McMillan, Ip, and Haller discuss the concepts and theory behind formal property checking, including an overview of property specification and a discussion of formal verification technologies and engines.

1.5 TEST

1.5.1 DESIGN-FOR-TEST

Chapter 21 by Koenemann and Keller discusses the wide variety of methods, techniques, and tools available to solve design-for-test (DFT) problems. This is a sizable area with an enormous variety of techniques, many of which are implemented in tools that dovetail with the capabilities of the physical test equipment. This chapter surveys the specialized techniques required for effective DFT with special blocks such as memories, as well as general logic cores.

1.5.2 AUTOMATIC TEST PATTERN GENERATION

Chapter 22 by Cheng, Wang, Li, and Li 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.

1.5.3 ANALOG AND MIXED-SIGNAL TEST

In Chapter 23, Stratigopoulos and Kaminska first overview 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 are reviewed and compared with the digital test. This chapter concludes Volume 1 of *Electronic Design Automation for Integrated Circuits Handbook*, Second Edition.

1.6 RTL TO GDSII OR SYNTHESIS, PLACE, AND ROUTE

1.6.1 DESIGN FLOWS

The second volume, *Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology* begins with Chapter 1 by Chinnery, Stok, Hathaway, and Keutzer. The RTL to GDSII flow has evolved considerably over the years, from point tools bundled 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.

1.6.2 LOGIC SYNTHESIS

Chapter 2 by Khatri, Shenoy, Giomi, and Khouja 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.

1.6.3 POWER ANALYSIS AND OPTIMIZATION FROM CIRCUIT TO REGISTER-TRANSFER LEVELS

Power has become one of the major challenges in modern IC design. Chapter 3 by Monteiro, Patel, and Tiwari provides an overview of the most significant CAD techniques for low power, at several levels of abstraction.

1.6.4 EQUIVALENCE CHECKING

Equivalence checking can formally verify whether two design specifications are functionally equivalent. Chapter 4 by Kuehlmann, Somenzi, Hsu, and Bustan defines the equivalence-checking problem, discusses the foundation for the technology, and then discusses the algorithms for combinational and sequential equivalence checking.

1.6.5 DIGITAL LAYOUT: PLACEMENT

Placement is one of the fundamental problems in automating digital IC layout. Chapter 5 by Kahng and Reda 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.

1.6.6 STATIC TIMING ANALYSIS

Chapter 6 by Cortadella and Sapatnekar 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.

1.6.7 STRUCTURED DIGITAL DESIGN

In Chapter 7, Cho, Choudhury, Puri, Ren, Xiang, Nam, Mo, and Brayton cover the techniques for designing regular structures, including data paths, programmable logic arrays, and memories. It extends the discussion to include regular chip architectures such as gate arrays and structured ASICs.

1.6.8 ROUTING

Routing continues from automatic placement as a key step in IC design. Routing creates the wire traces necessary to connect all the placed components while obeying the process design rules. Chapter 8 by Téllez, Hu, and Wei discusses various types of routers and the key algorithms.

1.6.9 PHYSICAL DESIGN FOR 3D ICs

Chapter 9 by Lim illustrates, with concrete examples, how partitioning the blocks of an IC into multiple chips, connected by through-silicon vias (TSVs), can significantly improve wire length and thus both performance and power. This chapter explores trade-offs between different design options for TSV-based 3D IC integration. It also summarizes several research results in this emerging area.

1.6.10 GATE SIZING

Determining the best width for the transistors is essential to optimize the performance of an IC. This can be done both in a continuous setting, oriented toward full-custom or liquid library approaches, and in a discrete setting, for library-based layout and FinFET circuits. In Chapter 10, Held and Hu emphasize that sizing individual transistors is not very relevant today; the entire gates must be sized.

1.6.11 CLOCK DESIGN AND SYNTHESIS

Chapter 11 by Guthaus discusses the task of distributing one or more clock signals throughout an entire chip, while minimizing power, variation, skew, jitter, and resource usage.

1.6.12 EXPLORING CHALLENGES OF LIBRARIES FOR ELECTRONIC DESIGN

Chapter 12 by Hogan, Becker, and Carney 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.

1.6.13 DESIGN CLOSURE

Chapter 13 by Osler, Cohn, and Chinnery 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 ASICs is used and illustrated. Finally, issues such as power-limited design and variability are discussed.

1.6.14 TOOLS FOR CHIP-PACKAGE CO-DESIGN

Chip-package co-design refers to design scenarios, in which the design of the chip impacts the package design or vice versa. In Chapter 14, Franzon and Swaminathan discuss the drivers for new tools; the major issues, including mixed-signal needs; and the major design and modeling approaches.

1.6.15 DESIGN DATABASES

The design database is at the core of any EDA system. While it is possible to build a mediocre EDA tool or flow on *any* database, efficient and versatile EDA tools require more than a primitive database. Chapter 15 by Bales 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.

1.6.16 FPGA SYNTHESIS AND PHYSICAL DESIGN

Programmable logic devices, 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. Chapter 16 by Hutton, Betz, and Anderson gives an overview of relevant architectures, CAD flows, and research.

1.7 ANALOG AND MIXED-SIGNAL DESIGN

1.7.1 SIMULATION OF ANALOG AND RF CIRCUITS AND SYSTEMS

Circuit simulation has always been a crucial component of analog system design and is becoming even more so today. In Chapter 17, Roychowdhury and Mantooth provide a quick tour of modern circuit simulation. This includes 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.

1.7.2 SIMULATION AND MODELING FOR ANALOG AND MIXED-SIGNAL INTEGRATED CIRCUITS

Chapter 18 by Gielen and Phillips 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.

1.7.3 LAYOUT TOOLS FOR ANALOG ICs AND MIXED-SIGNAL SoCs: A SURVEY

Layout for analog circuits has historically been a time-consuming, manual, trial-and-error task. In Chapter 19, Rutenbar, Cohn, Lin, and Baskaya 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.

1.8 PHYSICAL VERIFICATION

1.8.1 DESIGN RULE CHECKING

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 (DRC) is to achieve high overall yield and reliability. Chapter 20 by Todd, Grodd, Tomblin, Fetty, and Liddell gives an overview of DRC concepts and then discusses the basic verification algorithms and approaches.

1.8.2 RESOLUTION ENHANCEMENT TECHNIQUES AND MASK DATA PREPARATION

With more advanced IC fabrication processes, new physical effects, negligible 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. In Chapter 21, Schellenberg discusses the compensation schemes and mask data conversion technologies now available to accomplish the new design for manufacturability (DFM) goals.

1.8.3 DESIGN FOR MANUFACTURABILITY IN THE NANOMETER ERA

Achieving high-yielding designs in state-of-the-art IC process technology has become an extremely challenging task. DFM includes many techniques to modify the design of ICs in order to improve functional and parametric yield and reliability. Chapter 22 by Dragone, Guardiani, and Strojwas 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.

1.8.4 DESIGN AND ANALYSIS OF POWER SUPPLY NETWORKS

Chapter 23 by Panda, Pant, Blaauw, and Chaudhry 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 electromigration.

1.8.5 NOISE IN DIGITAL ICs

On-chip noise issues and their 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 by Keller and Kariat in Chapter 24, gained in recent years.

1.8.6 LAYOUT EXTRACTION

Layout extraction is the translation of the topological layout back into the electrical circuit it is intended to represent. In Chapter 25, Kao, Lo, Basel, Singh, Spink, and Scheffer discuss the distinction between designed and parasitic devices and also the three major parts of extraction: designed device extraction, interconnect extraction, and parasitic device extraction.

1.8.7 MIXED-SIGNAL NOISE COUPLING IN SYSTEM-ON-CHIP DESIGN: MODELING, ANALYSIS, AND VALIDATION

Chapter 26 by Verghese and Nagata 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.

1.9 TECHNOLOGY COMPUTER-AIDED DESIGN

1.9.1 PROCESS SIMULATION

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. In Chapter 27, Johnson discusses the history, requirements, and development of process simulators.

1.9.2 DEVICE MODELING: FROM PHYSICS TO ELECTRICAL PARAMETER EXTRACTION

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. Chapter 28 by Dutton, Choi, and Kan starts with the physical description of IC devices and describes the evolution of TCAD tools.

1.9.3 HIGH-ACCURACY PARASITIC EXTRACTION

Chapter 29 by Kamon and Iverson 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 (Springer), Dordrecht, the Netherlands, 2002.