Limitations and Challenges of Computer-Aided Design Technology for CMOS VLSI

RANDAL E. BRYANT, FELLOW, IEEE, KWANG-TING CHENG, FELLOW, IEEE, ANDREW B. KAHNG, KURT KEUTZER, WOJCIECH MALY, FELLOW, IEEE, RICHARD NEWTON, MEMBER, IEEE, LAWRENCE PILEGGI, SENIOR MEMBER, IEEE, JAN M. RABAEY, FELLOW, IEEE, AND ALBERTO SANGIOVANNI-VINCENTELLI, FELLOW, IEEE

Invited Paper

As manufacturing technology moves toward fundamental limits of silicon CMOS processing, the ability to reap the full potential of available transistors and interconnect is increasingly important. Design technology (DT) is concerned with the automated or semi-automated conception, synthesis, verification, and eventual testing of microelectronic systems. While manufacturing technology faces fundamental limits inherent in physical laws or material properties, design technology faces fundamental limitations inherent in the computational intractability of design optimizations and in the broad and unknown range of potential applications within various design processes. In this paper, we explore limitations to how design technology can enable the implementation of single-chip microelectronic systems that take full advantage of manufacturing technology with respect to such criteria as layout density, performance, and power dissipation. One limitation is that the integrated circuit (IC) design process—like any other design process—involves practical tradeoffs among multiple objectives. For example, there is a need to design correct and testable chips in a very short time frame and for these chips to meet a competitive requirement. A second limitation is that the effectiveness of the design process is determined by its context—the design methodologies and flows we employ, and the designs that we essay—perhaps more than by its component tools and algorithms. If the methodology constrains the design in a particular way (e.g., row-based layout, or clocked-synchronous timing), then even if individual tools all perform "optimally," it may be impossible to achieve an optimal result. On the other hand, without methodological constraints there are too many degrees of freedom for developers of design technology to adequately support the designer. A third limitation is that while the design process as a whole seeks to optimize, the underlying optimizations are computationally intractable. Hence, heuristic approaches with few if any guarantees of solution quality must be ever-present within design technology. This is perhaps the sole "fundamental limit" in design technology.

Design technology by itself does not impose any fundamental limits on what can be implemented in silicon. And while "optimal use of silicon technology" is an ill-posed objective (going far beyond the scope of algorithms, tools, methodologies, and infrastructure), design technology is the key to approaching and realizing the limits imposed by other aspects of the design process. In this paper, we summarize the mainstream methodologies used by CMOS silicon designers today and—against the backdrop of International Technology Roadmap for Semiconductors (ITRS) forecasts—point out basic limitations in their ability to achieve "optimal" design quality using reasonable resources. In each area of today's mainstream design flow, we either identify and quantify the factors limiting progress or point out the work that must be done to obtain such an understanding. In particular, we emphasize the role of metrics in the design process and how we might establish them. Finally, we present a number of potential solutions to these problems in the form of methodological approaches and major outstanding research questions that are being considered actively within the design technology research community.

Keywords*—*CMOS digital integrated circuits, Design automation, design for testability, design methodology, very large-scale integration.

Manuscript received March 2, 2000; revised October 19, 2000.

R. E. Bryant is with the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213-3891 USA (e-mail: Randy.Bryant@cs.cmu.edu).

K.-T. Cheng is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560 USA (e-mail: timcheng@ece.ucsb.edu).

A. B. Kahng is with the Computer Science and Engineering Department, and the Electrical and Computer Engineering Department, University of California at San Diego, La Jolla, CA 92093-0114 USA (e-mail: abk@ucsd.edu).

K. Keutzer, R. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli are with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720 USA (e-mail: keutzer@eecs.berkeley.edu; newton@coe.berkeley.edu; jan@eecs.berkeley.edu; alberto@eecs.berkeley.edu).

W. Maly and L. Pileggi are with the Electrical and Computer Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213-3891 USA (e-mail: maly@ece.cmu.edu; pileggi@ece.cmu.edu).

Publisher Item Identifier S 0018-9219(01)02070-9.

I. INTRODUCTION

Design technology (DT) comprises *algorithms*, software and hardware *tools*, and *design methodologies* (manifest as *design flows*) that are used for the efficient *conception, implementation, verification*, and *testing* of microelectronics-

0018–9219/01$10.00 © 2001 IEEE

based systems. Aspects of design technology have been referred to as electronic computer-aided design (ECAD), electronic design automation (EDA), and high-level design automation (HLDA). We use the term *design technology* comprehensively, encompassing all of these activities as well as newer ones that are evolving rapidly today. Without design technology, it would be impossible to implement, verify, and test the complex, single-chip electronic systems that are the foundation of today's information technology revolution. It is through design technology that the ideas and objectives of the electronic systems designer are transformed into reality; the quality of the design tools and associated methodologies determine the design time, performance, cost, and correctness of the final system product.

In today's highly competitive environment, even small differences in the quality of one design flow versus another can be the difference between success and failure, and a major improvement can lead to an entirely new generation of commercial tools and services. Thus, it would be very useful to know how close a given piece of design technology is to its "*fundamental limits*" of performance, e.g., as a synthesis, verification, or test system. Unfortunately, the question of determining such limits lies somewhere between ill-posed and intractable: we can only identify "*fundamental limitations*" of design technology. First, a metric for design technology is difficult to define. As with other types of technology, it may be that as one approaches any single fundamental limit in design technology, one achieves a monotonically better outcome in some dimension (say, smaller die area or lower power dissipation). However, actual design problems involve tradeoffs and a multivariate objective, e.g., minimizing design time may be a key objective in today's economy, but a rapidly implemented design that costs ten times more than it might otherwise, or that is delivered with subtle errors due to incomplete verification, may not be "better." Finding the best tradeoff among such parameters as design time, cost, power dissipation, and performance is a complex, situation-dependent process, and indeed the notion of *tradeoff* (power versus area versus speed; solution quality versus runtime; etc.) is at the core of design technology. Second, design technology is always applied within an external context that has an impact on its effectiveness. An individual tool or algorithm may be applied in the context of an ill-chosen methodology or flow within which even an "optimal" output from the individual tool cannot lead to an "optimal" overall result. Or, a design specification created by a human may be ill-considered (say, with respect to architecture or circuit design) or even unrealizable: since design technology is merely an amplifier of human endeavor and creativity, it can hardly guarantee "optimal use of silicon" under such circumstances. Finally, a third limitation is that while manufacturing technology seeks to *make* or *instantiate*, design technology seeks to *optimize*. Even ignoring issues of process, human factors, etc., we find that the underlying optimizations such as graph partitioning, multicommodity flow, scheduling, or quadratic assignment are almost always intractable, i.e., NP-hard [84]. Indeed, for certain classes of difficult optimizations, including many that arise in today's IC design process, no constant-factor approximation algorithm can exist unless the complexity classes P and NP are equal. Since NP-hardness may be interpreted to mean that no efficient optimal algorithms will likely ever be found, heuristic approaches are ever present within design technology. This is perhaps the main "fundamental limit" in design technology.

In the remainder of this section, we discuss two concepts that are at the heart of our vision for design technology and its future. First, we discuss the mechanisms by which tools and methodologies *coevolve* in step with process technology characteristics and design challenges. Many design challenges arise from "problems of the large"—the system complexities that result from smaller geometries and more transistors per die. Other challenges arise from "problems of the small"—the complexities of timing, signal integrity, power, manufacturing variability, and yield, etc., that result from process scaling into the deep-submicrometer (DSM) regime. The design challenges that arise in DSM processes, and how they impact today's design tools and methodologies, comprise our first area of focus. Second, we discuss the concept of *levels of abstraction* of design specification and description and in particular on the natural demarcation between *architecture* and *microarchitecture*. Today's design technology has mostly addressed implementation of microarchitecture; our second area of focus is the expanded scope of design technology that is necessitated by "problems of the large" and the need to maintain design (and silicon) productivity.

A. Tools Versus Methodology

As we make progress in design technology, there is an ongoing debate within the design technology community about what is more important: *new algorithms and tools*, or *new methodologies and associated tool flows*. What should come first: a breakthrough via a new algorithmic approach, usually manifest in a tool—or a basic change in the way the design problem is formulated, motivated by changes in technology or complexity, or by changes in overall design objectives? Where will the maximum benefit be gained and how close can it come to the best possible situation? The simple fact is that in the history of design for microelectronic systems, the answer has always been "both," and, in fact, these two aspects of the field of design technology are tightly coupled and highly correlated in terms of impact. At the same time, as silicon technology marches forward, chips of exponentially higher complexity are developed (following Moore's Law) but are based on a technological foundation whose characteristics are evolving rapidly as well (more interconnect levels, faster but weaker gates, increased power dissipation and noise concerns, greater manufacturing variability, etc.). According to the 1999 ITRS, the combination of exponential design complexity in sheer numbers of transistors and wires, along with the explosion in the number of design concerns, leads to "superexponential" growth in the complexity of the design task. To meet these new silicon technology contexts, the design methodology must change dramatically and its tools must be developed *in anticipation of* these changes. This situation is illustrated schematically in Fig. 1.

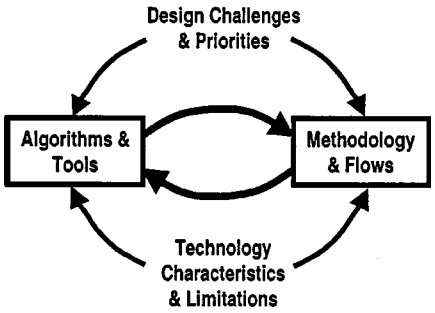

Fig. 1. Coevolution of tools and methodology with silicon technology and design challenges.

The "*problems of the large*" (increasing design complexity and the scale of the application), as well as the "*problems of the small*" (the ever-changing impact of physical laws, material properties and circuit innovations on what design can achieve, and with what level of effort) create a rapidly evolving context for design technologists and designers alike. A given design flow, tool, or even algorithm cannot continue indefinitely to be sufficient for design in the face of such changes. Every once in a while, we reach a point in technology where we cannot continue to "patch" the old tool or the old approach to design and must start again, almost from scratch. It is our conjecture that we are at such a point today, for reasons developed below. Moreover, as noted above, the limitations of what can be achieved by design are very much a function of both tools and methodology, as well as the designer's priorities in terms of complex tradeoffs among many objectives. Therefore, a key responsibility of design technologists today is to provide an objective quantification of the quality of tools and associated methodologies, delineating not only how well they perform relative to alternative approaches but how close they come to any type of fundamental limit inherent to the silicon technology. It is also essential to present to designers as accurate a picture as possible of available tradeoffs in the various dimensions of a given design. To this end, we note that useful metrics are very difficult to develop and calibrate. Since the commercial world of design and design technology is very competitive, issues of proprietary intellectual property (IP) present a large barrier. In addition, the complexity of the various tradeoffs mentioned earlier make simple metrics almost useless.

B. Levels of Design Abstraction

In the design of a single-chip electronic system, there are a number of levels of abstraction at which the design is likely to exist in the course of its evolution from an idea or a specification to a physical artifact. We use the taxonomy presented in Fig. 2 to describe these different levels. For the complex, system-on-chip (SOC) designs of today, designers typically begin with a *behavioral* specification of what they want to build. This description, or specification, expresses the functionality the design must implement, along with a set of constraints it must meet to be practical (cost, performance, power or energy dissipation, size, etc.), but (in an ideal world) does not say anything about how the design should be implemented. For example, if one were designing a wireless transceiver, this specification might contain a Matlab description of the various algorithms used in the processing of the digital wireless input signal, along with maximum bit-error-rate requirements, power dissipation requirements, and a cost target. Whether the best way to implement the design is as software on a digital signal processor (DSP) or as a hardware-only, application-specific IC is not at issue at this stage of the design. In most design approaches, the next stage of the design process involves the evaluation of tradeoffs across what we refer to as the architecture/microarchitecture boundary. While the word architecture is used in many meanings and contexts, we adhere to the definitions put forward in [24]: the *architecture* defines an interface specification that describes the functionality of an implementation, while being independent of the actual implementation. The *microarchitecture*, on the other hand, defines how this functionality is actually realized as a composition of modules and components, along with their associated software. The instruction-set architecture (ISA) of a microprocessor is a good example of an architecture: it defines what functions are supported by the processor without defining how these functions are actually realized. The microarchitecture of the processor is defined by the "organization" and the "hardware" of the processor [25]. These terms can easily be extended to cover a much wider range of implementation options. At this point, the design decisions are made concerning what will eventually be implemented as software or as hardware.

Today, most VLSI hardware design flows begin with a register-transfer level (RTL) description in either the VHSIC hardware description language (VHDL) or Verilog. The description is transformed by logic synthesis to a logic-level structural representation (a gate-level netlist, consisting of logic gates, flip-flops, latches, etc., and their interconnections) and then via layout synthesis tools (floorplanning, placement, and routing) to a final physical layout that is ready for transfer to manufacturing. This latter part of the flow is presented in more detail in Section III.

C. Scope and Outline of the Paper

Many different tool collections and design flows are used in the design of today's complex processors and SOCs. However, the majority of the circuits designed today, and certainly the majority of the transistors implemented, are designed using a variation of the flow introduced in detail in Section II. While an SOC today presents significant challenges due to its heterogeneity (possibly containing analog, radio-frequency (RF), mixed-signal, photonic, and even microelectro-mechanical components), space limitations prevent us from doing justice to the limitations and challenges associated with these aspects of the design problem. We place our emphasis on the design of the complex, digital portions of such systems. In particular, we believe that the extension of design methodology to higher levels of abstraction as well as improving the predictability of the design of increasingly

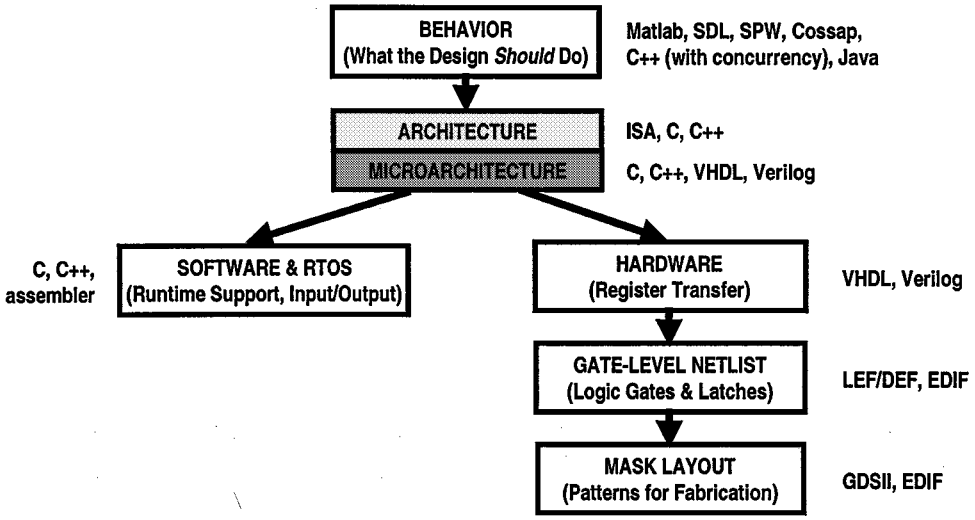

Fig. 2. Levels of design representation and sample associated formats.

complex digital systems present a major challenge and are representative of the kinds of issues we will have to accommodate.

We have organized this paper into three major sections. In Section II, we review today's mainstream, synthesis-based approach to the design and implementation of mostly clocked-synchronous complex digital integrated circuits. In Section III, we point out a number of fundamental factors that prevent designers from achieving "optimal" design quality or silicon utilization in a reasonable time. We base our discussion on our understanding of limitations in *today's* approach to design as well as the predicted directions for silicon manufacturing technology as expressed in the International Technology Roadmap for Semiconductors (ITRS) [11]. In each area of the design flow, we either identify and quantify the factors limiting progress or point out the work that must be done to obtain such an understanding. In particular, we emphasize the role of metrics in the design process and how we might establish them. Section IV proposes a number of approaches—requiring new methodologies as well as new algorithms and tools—for the reliable and efficient design of such systems. These innovations are under active consideration within the design technology research community (e.g., [12]). For each methodology, we describe major outstanding research problems that must be addressed by the design technology community. Only through such new approaches to design can we hope to approach the fundamental limits of silicon technology. Finally, we note that the distinction between "today's challenges" (Section III) and "tomorrow's approaches" (Section IV) is never a clean one; some overlap and arbitrariness in the partitioning is inevitable.

II. Today's Mainstream (Synthesis-Based) Design Methodology

While there are many different approaches to the use of design technology for IC design, most of the transistors we use today are designed with a mainstream design flow, as illustrated in Fig. 3. This flow has certainly evolved over the years as new tools have been added to the design methodology, but the major elements of the flow have remained unchanged since the late 1980s. The flow implements a *clocked*, *synchronous* design style, where the entire design is temporally partitioned into a collection of combinational subnetworks using a clock signal. As shown, it represents a slightly more detailed expansion of the bottom three abstraction levels given in Fig. 2. While this is but one of many possible design methodologies, it is by far the most commonly used today.

In this flow, the designer begins with a description of the digital functionality desired in the implementation. This description is expressed in a register-transfer level hardware design language (HDL), such as VHDL or Verilog. Via a *sequential synthesis* step, this HDL description is optimized and mapped to a more explicit register-level description, where the syntax more closely represents the final design structure (hence the term RTL netlist). *Logic optimization* and *technology mapping* transform this description to a logic-gate-level representation (logic netlist) using the logic gates and storage elements (flip-flops, latches) from a given library of such elements (e.g., the standard-cell library of a foundry). Finally, these gate-level logic elements are placed in a pattern and then interconnects (wires) are added to the layout, in a step often referred to as *physical synthesis* or *layout synthesis*. After this step, the design is ready to be passed to mask and manufacturing processes.

III. Silicon Technology Trends and Their Implications for Design Technology

In a world where major characteristics are changing at an exponential rate, one cannot go for very long without something reaching its breaking point. The overall VLSI design methodology described in Section II has changed little over the past decade, and we believe that major revolutions in methodologies and associated tools are essential in order to meet the productivity and reliability requirements of the IC design com-

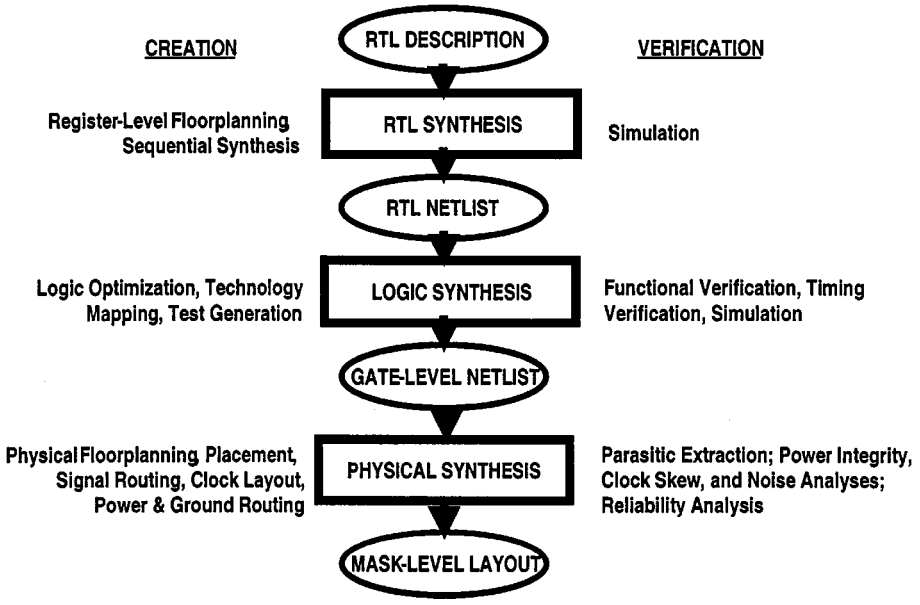

Fig. 3. Basic RTL-to-mask flow for VLSI design.

munity. Impetus for this change comes from: 1) the economic trends expressed in the ITRS with respect to manufacturing and validation costs and 2) the physics of DSM devices and interconnects. We make the following observations.

First, the cost of a state-of-the-art microelectronics fabrication facility continues to rise, with a new 0.18-μm high-volume manufacturing plant now costing approximately \$2–\$3 billion. This cost is driving a continued consolidation of the back-end manufacturing process, with many vertically integrated device manufacturing companies either partnering with pure-play silicon manufacturing partners (e.g., TSMC, UMC today) or moving to an entirely outsourced manufacturing model. Increasing cost of manufacturing capability also prejudices manufacturers toward parts that have guaranteed high-volume production from a single mask set, or that are likely to have high-volume production if successful. Such a requirement translates to better foundry response time and higher prioritization when global manufacturing resources are in short supply.

Second, the nonrecurring engineering (NRE) cost associated with the design and tooling of complex chips is growing rapidly. The ITRS and its "red brick walls" reveal that while CMOS technology will remain feasible well into sub-50-nm minimum feature sizes, the production of practical masks and exposure systems will likely be a major bottleneck for the development of such chips. For example, even with a shift from 4× to 5× reduction systems (relaxing the manufacturing and inspection tolerances of the mask-making industry), the cost of masks will grow extremely rapidly for these fine geometries, increasing the NRE for a new design. Today, a single mask set and probe card costs \$1 million, ten times the cost of a decade ago. We estimate that NRE cost for design and manufacture of samples for a complex CMOS IC in a state-of the-art process is between \$7–\$10 million today. When manufacturing and NRE cost trends are viewed holistically in the ITRS context, a fundamental "cost contradiction" [96] is seen to be inherent in Moore's Law and the ITRS. This contradiction and its likely implications for the structure of the design–manufacturing interface are discussed in Sections III-D and IV-F below.

Third, design validation is now well understood to be the limiting factor in both time-to-market and (perhaps more important) the *predictability* of time-to-market. This is due to the increasingly significant effects of physics in modern fabrication processes (e.g., affecting on-chip communication, reliable power distribution—"Did I actually get what I designed?") as well as the impact of increasing design complexity ("Did I actually design what I wanted?"). As chips become more complex and take on more system functionality, one of the most difficult challenges lies not in modeling the behavior of the chip itself, but rather in modeling the behavior of the environment in which the chip is to be used. This increase in the *context complexity* of modern chip design problems significantly increases the value of being able to prototype the design in its actual final application, operating at real system speeds, before widespread deployment. The specification of the actual design requirement is often incomplete or is evolving over time; hence, incremental redesign and revalidation take on increased significance. Last, the cost of developing and implementing comprehensive tests will continue to represent an increasing fraction of total design cost unless new approaches are found.

Fourth, it is not only design complexity and chip cost that challenge future design technology. We also face "problems of the small": dimensions are becoming smaller, statistical process variations are becoming increasingly significant, and device and interconnect scaling trends make performance of the physical implementation unpredictable during "front-end" system-level design. Thus, another major challenge is achieving reliable and predictable system implementation from the microarchitecture level down to layout.

The remainder of this section reviews existing technologies and associated technical challenges for five key steps of the design methodology, following the sequence of Fig. 3. In order, these are: system-level design, functional design and implementation verification, timing closure (i.e., across logical and physical design), physical design and physical verification, and manufacturing test and analysis.

A. System-Level Design

Preferred approaches to the implementation of complex embedded systems will likely be affected by the following factors.

- *Design costs and time* are likely to dominate the decision-making process for system designers. Therefore, design reuse in all its shapes and forms will be of paramount importance. Flexibility is essential to be able to map an ever-growing functionality onto an ever-evolving hardware.
- *Designs must be captured at the highest level of abstraction* to be able to exploit available implementation degrees of freedom. Such a level of abstraction should not make any distinction between hardware and software, since such a distinction is the consequence of a design decision.
- Next-generation systems will use a few highly complex part types (at the limits of Moore's Law or the ITRS), with many more *energy-power cost-efficient, medium-complexity chips* [O(10–100 M) gates in 50-nm technology], working concurrently to implement solutions to complex sensing, computing, and signaling/actuating problems. That is, for most applications chip design will be driven by cost considerations far more than by limits of complexity.

In this context, chips will most likely be developed as instances of particular *platforms*. Rather than being assembled from a collection of independently developed blocks of silicon functionality, each will be derived from a specific "family" of microarchitectures, possibly oriented toward a particular class of problems, that can be modified (extended or reduced) by the system developer. These platforms will be extended mostly through the use of large blocks of functionality (for example, in the form of coprocessors), but they will also likely support extensibility in the memory/communication architecture.

A major limitation of today's mainstream design methodology is the level at which one must enter the design. While RTL-level design entry has been sufficient for the past decade, design complexity and the types of functionality that must be implemented in today's SOC designs have reached the point where designers are routinely starting from a higher level source for design description. Such sources include Matlab, SDL, C, or C++ (extended to support concurrency), and Java [22]. With the continued evolution of today's embedded systems market, the emphasis of such design entry systems must also comprehend the importance of data streaming (dataflow) representations, as well as the conventional control-oriented approaches that are most easily described using a collection of concurrent finite-state machines. At most embedded system design companies and IC design companies, designers work at levels of abstraction that are too close to implementation: most IC designers have an RTL language description as their highest level of abstraction, and most embedded system designers use assembly or at best C language to capture and implement the design. These levels are too low for complex system design, e.g., sharing design components and verifying designs before prototypes are built is nearly impossible. In general, designers are thwarted by the low productivity afforded by the expressive power of RTL languages by missing support for software implementations, and by the lack of appropriate modeling of concurrency in all its incarnations.

A design methodology that effectively addresses complex systems must start at high levels of abstraction. However, the *ad hoc* adoption of higher level abstractions that is happening today only goes partway in solving the problem. Higher level descriptions must be based on a well-defined model of computation; only then can the benefits in terms of complexity management, concurrency exposure, optimization, and verifiability be fully exploited.

1) Managing Concurrency as a Fundamental Limit: Perhaps the most significant technical challenge facing systems designers in the years ahead, and the issue that is most likely to fundamentally limit design productivity, is that of *concurrency*. When real-time data movement is a critical element of an application, when latency (due to interconnect delay, for example) dominates bandwidth, and when chips are sufficiently complex that they must handle multiple tasks at the same time to be cost effective, how one reliably and efficiently implements concurrency becomes of the utmost importance. In essence, whether the silicon is implemented as a single, large chip or as a collection of smaller chips interacting across a distance, the *problems associated with concurrent processing and concurrent communication must be dealt with in a uniform and scalable manner*. In any large-scale embedded systems program, concurrency must be considered as a first class citizen at all levels of abstraction and in both hardware and software. This is a problem whose solution has eluded the software community for 50 years but must be solved—both in hardware as well as in software—if we are to even approach the fundamental limits provided by silicon technology in the years ahead. This problem cannot be solved by a better tool or a faster algorithm alone. It requires a change in methodology—a change in the way we design concurrent systems to an approach rooted in one or more formal mathematical models that give us the foundation we need to guarantee certain properties of the design.

2) System-Level Design Reuse: As the complexity of the products under design increases, the development effort required increases exponentially. When addressing *problems of the large*, one of the few remaining productivity levers to exploit is the idea of design reuse. Clearly, this is not a new idea, e.g., at the printed circuit board level each packaged component on the board is an example of design reuse. In

many ways, application-specific IC (ASIC) design methodology regularly exploits reuse at the gate and medium-scale integration levels, where the ASIC library defines a collection of reusable, precharacterized components. And of course in the software world, design reuse has been a goal for many years, most recently promoted in the form of object-oriented design with reuse of classes, as well as the component-based design approach used most often to assemble user interfaces or in products like Microsoft Visual Basic.

Both reuse and early error detection imply that the design activity must be defined rigorously, so that all phases are clearly identified and appropriate checks are enforced. Design reuse is most effective in reducing cost and development time when the components to be shared are close to the final implementation. On the other hand, it is not always possible or desirable to share designs at this level, since minimal variations in specification can result in different, albeit similar, implementations. Higher level abstraction can eliminate the differences among designs, enabling sharing with only a minimal amount of work needed to achieve final implementation.

The ultimate goal is to create a library of functions and of hardware and software implementations that can be used for all new designs. It is important to have a multilevel library, since it is often the case that the lower levels that are closer to the physical implementation change because of the advances in technology, while the higher levels tend to be stable across product versions. Both system and software reuse imply a design methodology that leverages available existing implementations at all levels of abstraction. This would allow preexisting components to be assembled with little or no effort.

B. Functional Design and Implementation Verification

The most significant bottleneck and ultimately the limiting factor on time-to-market for a reliable SOC design is design verification. With increasingly complex designs, this problem is looming increasingly large and radical new approaches are essential. In this section, we summarize the state-of-the-art approaches to functional verification of complex digital systems and point out the key directions needed in research if we are to expand the limitations of the verification task.

1) Simulation-Based Approaches: Logic simulation is the dominant technology used by industry for functional verification. With this approach, multiple models are constructed describing the system at different levels of abstraction. Typically, a very high-level description might be written in a standard programming language, while more detailed ones are written in a hardware description language such as Verilog or VHDL. Simulation patterns are generated to exercise the system over a wide variety of operating conditions. These patterns are then simulated and the results are analyzed to determine whether the simulated model behaves as desired. This analysis generally involves making sure that the different models produced consistent results and that specific undesirable behaviors did not occur. In addition, one might check that specific desirable behaviors *did* occur to make sure that the patterns provided adequate coverage of the system's possible behaviors.

In the design of a state-of-the art system, enormous resources are expended on simulation-based verification. Companies set up entire "farms" consisting of several hundred computers that run simulations continuously. Large numbers of verification engineers—typically more than there are designers—create test patterns and analyze the results of simulation. As designs become larger and more complex, this expenditure of resources does not seem to scale favorably: simulation consumes an ever-larger portion of the computer and human resources in the design process and is often a major limiting factor in bringing a design to market.

Simulation-based verification has been remarkably effective at ensuring high-quality electronic designs. One need only look at the electronic systems available commercially to see that our industry meets much higher quality standards than do others, particularly the software industry. This high degree of success has led to high expectations, however. In 1994 when it was discovered that the Intel Pentium processor generated slightly incorrect results for floating point division in less than one out of every billion possible cases, Intel was obligated to mount a massive recall campaign costing $475 million. In today's marketplace where there are far more microprocessors deployed, many in embedded systems, the cost of such a recall is almost incalculable. Thus, there is very high pressure to make sure that electronic systems are free of functional defects.

Unfortunately, as systems become larger and more complex, simulation becomes less effective at finding possible design errors. This is particularly true for systems that have high degrees of concurrency, such when there are multiple agents performing their tasks independently, interacting and communicating over some medium such as a bus. For example, an automotive electronic system might have separate agents performing monitoring and controlling the engine, the automatic transmission, the antilock braking system, the driver's console, and the airbags. Some agents must interact continuously, such as the engine and transmission controls, while others interact only in special cases, such as when the brakes and transmission coordinate for traction control. In such systems, errors often occur under only unusual combinations of events, e.g., when the driver depresses the accelerator, the automobile senses and deploys traction control, and then the driver depresses the brake pedal within a short amount of time. Such event combinations are difficult for the designer to enumerate, making it hard to generate a comprehensive set of simulation tests. The number of possible event combinations can become far too large to simulate. Even determining whether the simulated behavior is correct can become difficult in highly concurrent systems, since we do not demand that it have a particular fixed functionality.

The EDA industry has developed a number of tools and techniques to support simulation-based verification. These can be classified as either making simulation faster or making it more effective for a given amount of simulation. To make simulation faster, most companies have recently shifted from event-based simulation in which each low-level component

is evaluated whenever one of its input values changes, to cycle-level simulation in which each component is evaluated once per clock cycle according to a fixed evaluation order. Cycle-level simulation can outperform event-based simulation by a factor of ten or more. In addition, more tools are available for managing simulations running on multiple machines, either to partition the simulation of a single simulation model across multiple machines or to run multiple copies of a simulation model over different simulation patterns. A final technique to improve simulation performance has been to map the simulation models onto special-purpose hardware, providing either a highly optimized simulation engine or to emulate system operation using programmable logic. These approaches tend to be less cost effective than running software simulators on conventional machines. Their use is mainly for creating models with sufficient performance and capacity to serve as a platform for software development or application evaluation. These efforts at improving simulator performance have been fairly successful, but we do not anticipate there will be comparable improvements in the future. Modern simulators running on conventional machines are highly optimized and efficient. Hardware acceleration and emulation will still satisfy the needs of only particular niches.

A more recent industry focus has been on making simulators more effective. Improvements include tools to generate simulation patterns that more fully cover the set of possible system behaviors, techniques to analyze the simulation results both for correctness and for coverage, and improved environments to support the activities of verification engineers.

The dominant technique for generating tests is to use some form of random pattern generation. For example, a microprocessor might be simulated using a sequence of random instructions. Random patterns have the advantage that they create many different event combinations, including ones that were often not anticipated by the designers. However, purely random patterns will often fail to generate the unusual event combinations that cause failures. With most random pattern generators, the user can bias them toward areas where difficulties often occur, e.g., to generate instructions causing a large number of exceptions or data hazards. A more recent technique has been to use techniques developed by the test industry for automatic test pattern generation to generate simulation tests that will cause some particular behavior. For example, a test generator might be called to create some known hazard condition that did not occur during any of the random simulations. These techniques are still fairly costly in terms of computational requirements and can often fail when dealing with large-scale systems.

A particularly difficult problem for simulation-based verification has been developing metrics to quantify the degree of coverage provided by a set of simulation patterns and to identify aspects of the system behavior that require greater testing. In purely numerical terms, one can easily argue that complete coverage by simulation is a hopeless task. Even a simple 32-b adder has over 10^{19} possible input combinations. A processor that could simulate 100 billion patterns per second would require over three years to simulate that many combinations. If we scale to a 64-b adder, the total simulation time becomes unimaginably large. Clearly, simulators do a better job of discovering errors in circuit designs than this purely combinatorial analysis would lead one to believe. Recently, the EDA industry has provided coverage analysis tools for logic simulators that adapt the "line coverage" metrics of the software industry. That is, they make sure that every line of code in the VHDL or Verilog model is evaluated at least once. While such a standard is clearly a minimal requirement for coverage, most users agree that it is not adequate. First, even if some line of code is evaluated, there is no guarantee that the result of that evaluation will reach any point in the circuit where its effect is observed. In the terminology of testing, it only ensures that a potential error is excited. More seriously, it measures only the occurrence of the individual events and not their interactions. Unlike software in which much of the complexity is in its sequential behavior and hence reaching a particular point in the code can be quite significant, hardware systems typically consist of fairly simple components having complex interactions. Code coverage fails to measure the occurrence of event combinations.

This lack of effective coverage metrics is a major impediment in making simulation more effective. As a simple example, the industry has no reliable means of knowing when it has performed enough verification to be able to declare a product ready for shipment. Most companies simply look at the number of errors that have been detected by simulation each day and declare their task as done once the number drops below some threshold for a long enough period. Without coverage metrics, there is no way to determine whether this decline in the number of detected errors is due to a low number of actual errors or that the additional simulation is not covering any new forms of behavior. Without coverage metrics, it is difficult for a novice verification engineer to learn his or her task or measure success. As a result, simulation remains somewhat of a "black art" that can take years to master.

C. Timing Closure

The timing closure problem is caused by the inability to predict interconnect loading on logic gates with adequate precision prior to physical design. Logic is optimized with respect to timing, power and area objectives with assumed values for interconnect load capacitances, but the actual values of these loads cannot be known until after layout. This chicken–egg impasse has in the past been broken by the use of *wireload models*, which statistically predict interconnect load capacitance as a function of signal net fanout, technology data, and legacy design information. At 0.5 μm and older technologies, logic synthesis could perform reasonably effective technology mapping, gate/cell sizing, and buffering using wireload models, since interconnect capacitance was a modest fraction of the total net capacitance. However, in DSM technologies the interconnect capacitance is a larger, sometimes dominant portion of the total net capacitance since device input capacitances scale downward while lateral intralayer interconnect capacitances scale upward much

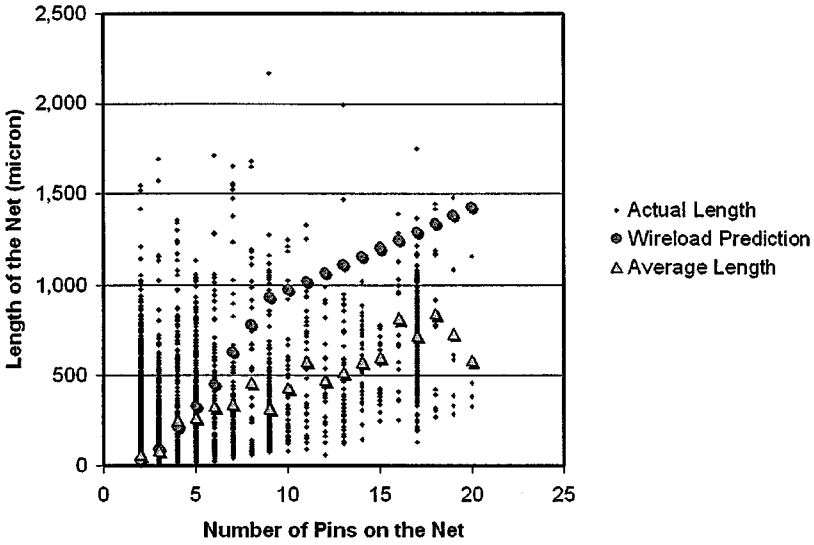

Fig. 4. Wirelength as a function of fanout for a modern design (source: Wilsin Gosti).

faster. Metal resistance effects also increase with routing length and are topology-dependent (i.e., dependent on where the routing tree branches are located, not just on how much wire is used in the tree); there is no simple statistical model to predict their impact on delay. Other basic reasons for the failure of wireload models in DSM are as follows. 1) The resistance–capacitance (*RC*) product per micrometer of IC wiring can vary by factors exceeding 100 depending on whether a connection is made on local layers (layers M1 and M2) or on global layers (layers M7 and M8 in recent processes); this means that detailed placement and routing information is needed for accurate parasitic estimates. 2) Interconnect estimates are used to validate designs in terms of worst case timing paths (critical paths) or other "max" criteria (by contrast, total wirelength is a "sum" criterion and is easier to predict). In this light, [85] uses order statistics arguments to explain why interconnect prediction (whether by wireload model or more sophisticated means) necessarily fails in the DSM regime. 3) With each process generation the increased system complexity and functionality causes average interconnect length to decrease less slowly than other physical dimensions. The result is an increase in the absolute number of outlying nets, i.e., nets that are not accurately captured in terms of a wireload capacitance model.

The above analyses imply that wireload models can be effective for a significant population of nets as scaling continues into the nanometer range, but can fail spectacularly for long and high-fanout nets [58], [60]. Fig. 4 shows the actual length (the first-order predictor for wiring capacitance and delay) for a modern 0.18-μm design as a function of the fanout (number of connected inputs) of each net. Not only does the wireload model fail to accurately follow the trend, but no simple formula could: wirelength is hardly correlated with fanout at all.

Of course, even if capacitance per unit wirelength does not scale with device input capacitance, there exists some net length for which device capacitance still dominates wire capacitance. This observation can be applied to a block-based chip design problem. Namely, for a given technology, one can predict the net lengths within the blocks will be small enough such that the timing estimates from statistical wireload models are sufficiently accurate [58]. It follows that current approaches for top–down synthesis can be applied to these block sizes. The corresponding interblock wires are by this definition considered long wires and remain problematic for such a block-based top–down design strategy. Unfortunately, long interblock nets often comprise portions of the critical timing paths. In the following, we review current approaches to timing closure and the open issues associated with each.

1) Placement-Aware Synthesis: To overcome the inability to predict delays via wireload models for certain block/design sizes, there has been a recent trend toward *placement-aware synthesis*. Placement-aware synthesis attempts to use a placement tool [46]–[48], [61] to perform a partial physical design to compute more accurate interconnect loading estimates on a net-by-net basis. Using this snapshot of placement information to remap logic immediately invalidates the physical design information, so incremental replacement is used in such approaches to iteratively refine such solutions. Convergence of such approaches is uncertain.

In addition to placement prediction, routing estimation is critical for accurate delay prediction of long DSM nets. Since delay prediction accuracy depends on both wiring lengths and layer assignments, placement information alone cannot capture all of the interconnect model detail, such as the increase in loading required to meander around a memory block or a clock tree spine. In addition, the increased coupling effects for DSM that are most problematic for long global nets cannot be predicted without some amount of routing detail.

2) Bounded Block Size for Logic Synthesis: Depending upon how one defines the scope of the tasks associated with logic synthesis, DSM trends may or may not require dramatic

change in the way *synthesis* is performed. For example, if synthesis algorithms are limited to blocks based on their own inherent complexity, then wireload models are likely to suffice. On the other hand, synthesis in the broader sense of the word, applied at chip level, will not be stretched easily into the DSM era.

There are two ways in which emerging technologies, and interconnect process scaling in particular, can require new synthesis methodologies. First, design styles and the multiperson nature of large IC design teams dictates the use of hierarchy in the modeling and analysis. If the hierarchical blocks are of a large enough size that interconnect effects become a factor, then layout-aware synthesis is warranted. Second, independent of how block synthesis is performed, an increasing number of components on a chip implies either that block sizes must increase correspondingly, or that the resulting block-assembly physical design problem will be quite difficult. In other words, if all blocks remain at the size for which interconnect is nondominant, this is at best a constant size (and at worst a decreasing block size). Therefore, the number of overall components (blocks) must be increasing, such that the number of interblock connections (now defined as wires for which interconnect can be dominant) explodes.

3) Time Budgeting Among Blocks: One way of addressing timing closure of hierarchical block-based designs is to allocate timing budgets to all of the subsystem components, including the global interconnects [49], [50], [52], [55], [57], [62]. The obvious problem is that reasonable delay prediction for global interconnects is difficult, if not impossible, before the design is physically constructed. In the best of circumstances numerous design iterations are required to achieve timing closure. Suboptimality arises if timing slack is wasted on regions that were overbudgeted due to insufficient physical design information. Meanwhile, other areas that were underbudgeted result in negative timing slack and overall failure to achieve timing closure. It should be emphasized that reallocating extra slack from one region to another over subsequent design iterations is an extremely difficult task for large designs. Each design iteration is quite costly, and the slack reallocation schemes can be oscillatory and/or nonconvergent. Unfortunately, the delay budgeting problem will only become more difficult as more wires become global wires whose pin-to-pin delays are strongly dependent on their actual implementation by detailed routing tools.

4) Constant Delay Synthesis: Instead of timing budgeting at the block level, elegant algorithms for finer grain delay budgeting have been proposed [3], [4]. These approaches rely on the notion that the "logical effort" (based on the fact that different gate topologies such as inverter or NOR have different inherent ability to drive capacitive load) should be equalized at all logic stages along any timing path [4]. For the case of no interconnect and no branching paths, the mathematics are elegant and the solutions are of comparable quality to hand-tuned design. The branching effects, however, are handled at the price of algorithmic elegance and solution quality. For long interconnects that dominate circuit performance, the constant-delay-effort derivation is further compromised since the constant-effort relies on a model of gates driving other gate capacitances only [4]. Therefore, dominant interconnect would seemingly impact the overall quality of the constant-delay synthesis results when considering general interconnect paths.

Technology trends [1] suggest that these negative impacts on quality will only worsen over time. For DSM it is apparent that interconnect capacitance alone will comprise an increasingly larger portion of a net's overall capacitance. For this reason constant delay synthesis seems to be an alternative to traditional synthesis only for problems that are accurately gauged in terms of wireload fanout models. Extending either of these synthesis methodologies to "global" wires seemingly requires physical design information, which precludes any possibility of *one-pass* synthesis success.

5) Wire Planning: Recognizing that without physical information one-pass convergence is unlikely, *combined* wire planning and constant delay synthesis was proposed in [58] and [51]. The methodology is based on a block-design style where the wires among blocks are planned, or constructed, and the remaining slack is allocated for the constant delay synthesis within the blocks. The difficulty with such an approach for DSM technologies is as follows. If blocks are of significant size, then the synthesis-with-dominant-interconnect problem remains. In contrast, if block sizes remain relatively small such that constant-delay synthesis with wireload models works, then blocks will eventually appear as point objects (sea-of-blocks) for gigascale integration. In this latter scenario, the majority of the wiring will be global wiring and the physical placement of the point-like blocks will be absolutely critical to the overall wire planning quality. If we further acknowledge the need to incorporate reused IP blocks, generated datapaths, etc., this corresponds to an extremely challenging physical design problem.

D. Physical Design and the Interface to Manufacturing

Physical design encompasses traditional steps of floorplanning, timing optimization, placement, and routing. These steps are increasingly intertwined with *physical verification*, which encompasses electrical rule checking (ERC), layout-versus-schematic checking (LVS), and design-rule checking (DRC), as well as resistance, inductance, and capacitance (*RLC*) parasitic extraction, delay calculation, and (static) verification of system timing and signal integrity. The accelerating technology roadmap, to first order, brings lower supply and threshold voltages, higher aspect-ratio local interconnects, and higher current and power densities along with exponentially rising design complexities. A number of DSM effects (signal delay, crosstalk-induced noise and delay uncertainty, inductance effects, substrate noise, etc.) make achieving speed–power performance goals extremely difficult. As a result, more *degrees of freedom* in circuit design are exploited. With interconnect design in particular, the number of available techniques is large: for on-chip global signaling, designers may exploit

shields/spacing/interleaving, wire tapering, driver/repeater insertion and sizing, low-swing differential-pair structures, etc. In this context, performance analysis and verification become key challenges within the design process: *static* and "filtering-based" methodologies must be applied to handle complexity, even as relevant phenomena such as crosstalk-induced signal delay are inherently *dynamic* (see, e.g., [70]). This results in increased *guardbanding* that leaves performance and value on the table to achieve a more tractable design process. Other "DSM effects" arise with respect to economics and with respect to manufacturability, reliability, and the design–manufacturing interface. These effects motivate restructuring of existing handoffs between physical design/verification and the mask and fabrication segments of the IC industry; they also motivate a need for design for manufacturability (DFM) across all levels of design abstraction. In this section, we first discuss consequences of device, interconnect, and supply scaling; we then discuss economic consequences and the design–manufacturing interface.

1) Coupling-Induced Timing Uncertainty: Crosstalk can affect the behavior of VLSI circuits in two ways: 1) incorrect functionality through introduction of noise at sensitive nodes; and 2) increasing (or decreasing) the interconnect delays. A major cause of timing (or delay) uncertainty is the increasing effect of crosstalk between parallel *RC* interconnect lines in DSM circuits [65], [66]. For DSM processes, the coupling capacitance can be as high as the sum of the area and fringe capacitance of a wire. A small sampling of available design methodologies for controlling coupling-induced logic and timing errors includes: 1) staggering clock arrival times at latches; 2) using shield wires; 3) increasing spacing between wires; 4) using repeater insertion to reduce noise sensitivity; 5) upper and lower bounding slew times to control coupling effects, etc. We require new quantified assessments of which design techniques are best—e.g., in terms of design time as well as chip-quality metrics such as speed, area, of yield—according to specific analysis techniques, and in specific contexts.

2) Clocking and Power Distribution: Further complications are expected for gigascale systems due to decreasing integrated-circuit dimensions and higher clocking frequencies. The implications are that synchronous systems, as we know of them today, will not be feasible in the future. Instead, designs will have some portions that are asynchronous, in terms of new global communication components, or quasi-synchronous, based on multicycle signal propagations. Timing verification must also handle IP cores and span the entire spectrum of design issues from interconnect effects to high-level timing and scheduling.

Control of power dissipation and power density is in many ways more daunting than timing. Power increases in direct proportion to functionality and operating frequencies; hence, voltages are scaled to the lowest limits permitted by noise margins and performance requirements. Voltage scaling, even when augmented by multithreshold and multisupply process technology, is not enough. Turning off unused portions of the system via gated clocking modes saves power, but it increases the complexity of the physical design and verification process.

Particularly at the early phases of design, prediction of timing and power dissipation is extremely difficult for gigascale systems since the design is obviously based on incomplete data. Any forecasted data must be bounded by confidence intervals that reflect the incompleteness of the design information. But providing such bounds remains a difficult problem at almost all design-abstraction levels. DSM technologies and the interconnect performance domination that comes with them creates new challenges for model abstraction and performance prediction. For instance, forecasting the impact of coupling prior to physical design and routing is currently not possible at the synthesis level, yet can make the difference in terms of timing closure.

3) Impact of Manufacturability on Physical Design and Verification Flows: In ultra-DSM processes of 100 nm and below, statistical fluctuations of dopant concentrations, thin-oxide-layer thickness, chemical–mechanical planarization of shallow-trench isolation and damascene local metal, etc., have enormous performance implications. Even today, cross-chip, cross-wafer, and cross-lot variations in feature dimensions depend not only on local geometric context (e.g., layout pattern density), but also on where in the reticle the line occurs and even which copy of the stepper was used (lens aberrations differ in each laser). In this context, even simple concepts such as "signal delay" can no longer be abstracted as numbers (possibly dependent on operating conditions, as noted above), but rather must be treated as distributions. Such numbers also have unprecedented correlations, e.g., 1) an overpolished (thin) M1 layer may imply a thick M1–M2 interlayer dielectric and taller via geometries; or 2) gates near the center of the die may have different speeds than gates near the boundary of the die [64], [78].

The heightened interdependencies between design and manufacturing are due in part to a fundamental crossover point in the evolution of VLSI technology. This crossover point occurs when minimum feature dimensions and spacings decrease below the wavelength of the light source. Pattern fidelity deteriorates markedly in this subwavelength lithography regime: to achieve desired critical dimension (CD) control, optical lithography must apply compensation mechanisms [69], [72]–[74], [77], [83] that either perturb the shape [via optical proximity correction (*OPC*)] or the phase [via phase-shifting masks (*PSM*)] of transmitting apertures in the reticle. The fundamental design problem introduced by OPC and PSM is that there is no longer any "isomorphism" between the layout design and the mask, nor between the mask and the fabricated silicon—although the layout and silicon must be as similar as possible. Yet another design–manufacturing link is introduced as more interconnect layers are required at each successive technology generation. For multilayer interconnects, we see a strong requirement for *planarized* processes that rely on chemical–mechanical planarization (*CMP*). Manufacturing steps involving CMP have varying effects on device and interconnect features, depending on local *density* characteristics of the layout pattern; hence, layout must be augmented

with "dummy" features to even out the layout density. This link between layout and manufacturability has grown in importance with the move to shallow-trench isolation and inlaid-metal processes [68], [81], [82].

Such design–manufacturing dependencies place burdens on the underlying design technology in unprecedented ways [67]. First, traditional physical verifications such as parasitic extraction and performance verification cannot be performed accurately without close understanding and modeling of, e.g., downstream dummy metal insertion. Second, mask manufacturing capabilities and costs must be carefully considered. For example, to control mask yield and cost, OPC and PSM insertion must understand that only some device or interconnect dimensions are worth the expense of careful enforcement—i.e., *functional intent* must be passed to traditional physical verification steps. Inserted OPC and PSM features must also be *inspectable* by optics-based inspection systems, and appropriate measures of downstream mask costs will additionally need to be applied at the layout level. Third, both PSM and CMP create *global* effects in layout verification: a given feature may affect the correctness (phase-assignability or surrounding layout density) of another feature thousands of micrometers away. This fundamentally differs from today's *local* context for DRC and requires the definition of new criteria for "design-rule correctness." Finally, thorny issues are raised by hierarchy and reuse (e.g., composability of PSM- or CMP-correct layouts in cell- or block-based design).

4) "Cost Contradictions" and Structure of the Design–Manufacturing Interface: Three root assumptions in the ITRS postulate: 1) exponential decrease in minimum feature size; 2) exponential decrease in memory size; and 3) exponential increase in "transistor density" for DRAMs, microprocessors (MPUs) and ASICs, with rates dictated by Moore's Law [95]. The ITRS also suggests that aggressive die size shrinks will be applied to achieve the necessary exponential decreases in cost per transistor/bit consistent with historical trends. From these assumptions one can derive [96] cost per chip trends and cost per layer per cm^2 trends: for future technology generations, cost per chip (computed this way) increases while cost per layer per cm^2 remains constant. Hence, at some point the cost of a die may be too high to be acceptable by the high-volume market. Since decreasing the cost per layer per cm^2 seems difficult in light of the complexity of next-generation lithography and defect control techniques, we arrive at a fundamental "*die cost—manufacturing cost contradiction*" [96].

These contradictory cost pressures suggest either, or a mix, of two basic scenarios within the IC industry. In one scenario, cost per transistor/bit could be kept at ITRS-mandated levels, but this may lead to a shortage of R&D resources required for development of new processes and equipment. Alternatively, the cost per transistor/bit restriction could be relaxed, allowing cost per die to increase, but this may have negative impact on the volume of ICs produced and sold. We believe [96] that (outside of forcing acceptance by the market of higher chip prices) the IC industry will therefore be compelled to improve, as aggressively as possible, manufacturing efficiency in three fundamental ways: 1) by seeking market positions that allow increase of manufacturing volumes; 2) by attempting to decrease the variety of processes and products; and 3) by practicing *design for manufacturability* (DFM) at all levels of design abstraction. Approaches 1) and 2) are consistent with the emergence of programmable platform-based design, as detailed in Section IV-B below. Approach 3) suggests new structure in the interface between design and manufacturing domains, which we will describe in Section IV-F.

E. Manufacturing Test and Analysis

DSM technology and high integration of system-on-a-chip designs are challenging test in a number of areas. The test community must cope with an enormous spectrum of difficult problems ranging from, for instance, high-level test synthesis for component-based design to noise and power dissipation problems in extremely high-performance (in reality analog) pin electronics. The new test challenges come from three different sources: 1) automated test equipment (ATE) technology; 2) design technology; and 3) semiconductor technology. In the following we briefly discuss the impact imposed by the trends in these three areas.

1) Automatic Test Equipment is Reaching Limitations in Testing High-Performance Devices: On-chip clock speeds are increasing dramatically while the tester overall timing accuracies (OTA) are not. The ITRS roadmap predicts that ATE accuracy will improve marginally from 200 to 175 ps by 2012 while the clock period of high-speed IC can reach 330 ps by 2012. This translates into increasing measurement errors that it could reach over 50% of the clock period by 2012. This trend implies increasing yield loss (estimated to be up to 48% by 2012) as guardbanding to cover tester errors results in the loss of more and more good chips.

The volume of test data per gate is expected to remain constant (up to about 1 kB test data per gate), which implies that much more data need to be stored in the tester and also passed across the chip boundary (about 10 GB test data for a ten-million-gate design), resulting in capacity and bandwidth problems. Furthermore, long test application time and excessive cost of ATE also contribute to the increasing overall test cost (and hence system cost).

2) Integration of Complex, Heterogeneous Components in SOC Poses New Test Problems: A key challenge of testing embedded components and component-based SOCs is the heterogeneity of the components along several dimensions: the components can be microprocessors, DSPs, multimedia, telecommunication, or interface components; either soft, firm, or hard cores; either digital or analog; and either synchronous or asynchronous. The diversity of the components makes different choices of component-level test strategies, like boundary scan, automatic test pattern generation (ATPG), or built-in self-test (BIST), more appropriate for each different type of component, necessitating a system-level test strategy that can incorporate all the diverse component-level test schemes. The gigascale system-level integration, while benefiting the system design process in several ways, makes accessibility and diagnosability of

deeply embedded components a major problem. Moreover, the configurability and customizability of system components available in the SOC design paradigm, like processors, peripheral components, and the underlying bus, memory, and communication architectures, make SOC testing using standard test and access mechanisms a challenge.

Cost-effective methods to test embedded analog, mixed-signal, and RF components and devices using multiple technologies, multiple logic families, and even multiple voltage levels are not available for such heterogeneous chips. A significant fraction of systems-on-silicon chips are mixed-signal designs due to the natural trend of incorporating the analog interface circuitry into digital chips. Even though for most mixed-signal high-end ICs, the analog circuitry accounts for only a small fraction of the total silicon, the cost to produce such mixed-signal devices is being dominated by their analog costs. It has been predicted by several sources that the ratio of analog testing cost to the total mixed-signal product cost will continue to increase if change to analog testing is not made. Currently, test evaluation, design-for-test, and self-test solutions for analog blocks and converters remain very limited. Most of the existing solutions are primarily based on functional testing and have frequency limitations. There are no standard figures of merit for evaluating analog test program and there exists no systematic test strategy for RF circuits. Developing cost-effective test methodologies and techniques for analog and mixed-signal components is therefore an essential part of the task for devising a complete solution for testing heterogeneous SOCs.

Also, conventional issues of overhead caused by design-for-testability circuitry will be replaced by new problems. For example, wires will cost more than gates, both in silicon area and delay. Because Moore's law shows no sign of abating, it is vital that the test solutions are able to scale as design size increases, which is not the case for many of today's test solutions.

3) Deeper Submicrometer Technology Invalidates Existing IDDQ Test Solutions and Causes New Failure Modes: For deeper submicrometer technology, the background quiescent current to ground (IDDQ) increases inexorably and the spread of IDDQ distribution is also increasing. IDDQ testing must be adapted to exist in an environment of decreasing signal to noise ratio, or be replaced with a better suited method that maintains its effectiveness in defect screening and reliability prediction.

Geometries shrink year by year while the defect sizes do not shrink proportionally. Furthermore, the increase of wiring levels and the increasing dominance of wire delay demand new fault models. Furthermore, many of today's signal integrity problems (which are targets of DSM design validation) are becoming test problems as well. Examples are distributed delay variations, crosstalk induced delay, and logic errors, excessive voltage drop and/or swing on power nets, and substrate and thermal noise. The effects of these noise sources to the product quality remain unknown, while it is becoming clear that process variations are now more likely to cause devices to marginally violate the performance specifications. Testing must target not only spot defects but also such parametric performance failures. A new class of "noise" faults caused by above-mentioned DSM parametric variations needs to be properly modeled to the levels of abstraction higher than the present electrical, circuit, and transistor levels, to support applications in fault simulation, test generation, design for testability, and built-in self-test.

4) Paradigm Shifts: Innovative test solutions are needed to address the challenges outlined above. Without new test methods, testing may become a key showstopper that limits future design and manufacturing technologies. Test technology must be able to support higher level design and test handoff. Testability must be analyzed and test circuitry must be inserted early in the design process of chips and systems at high level of design abstraction. To overcome ATE limitations, the ATE systems must become simpler and ICs must be capable of more self-testing. Finally, manufacturing tests need to target a new class of parametric noise faults in digital DSM devices, and test tools must be able to generate high-quality tests and insert necessary DFT circuitry to identify such hard-to-detect defective devices.

IV. Emerging Approaches in DSM Design

In the previous section, we have outlined the major standard approaches in use today for the design, verification, and test of complex single-chip digital systems. We have also pointed out a number of the most significant limiting aspects of such approaches. In the following, we describe a number of the highest potential approaches—methodologies, tools, and algorithms *and infrastructure*—that we believe can ameliorate these limitations and extend our ability to exploit the fundamental limits of silicon technology.

A. Design Reuse and Component-Based Design

One of the major advances in software over the last few years has been the widespread and growing appreciation for component-based application development. In fact, many would argue that in the commercial world component-based software construction is proving considerably more effective than object-oriented techniques for a wide class of applications programming. In a component-based approach, there are two major phases in the design of an application: the *component development phase* and the *component assembly phase.*

During component development, the components themselves are constructed by experts who have expertise both in the domain of the component (e.g., graphics, user interface, signal processing) as well as in the construction of efficient, object-oriented software. *For the development of the component, the quality and efficiency of the tools (e.g., compilers, debuggers) and the development environment are essential to success.* In the component development world, there is a significant amount of reuse via objects and associated class libraries, but the most useful classes are usually very low-level interfaces to system services. For reasons of optimization and the need for sufficiently useful interfaces, much of the code

in a component is developed from scratch and is particular to a relatively small group of components.

In an object-oriented approach to software development, the application developer then assembles objects to build a particular application *using the same development environment that was used to develop the underlying classes themselves*. That is, the application *developer must be as much an expert in object programming as the developer of the underlying classes*. This is where the component-based approach differs significantly from its object-programming alternative. In an ideal component world, *the assembly of components can be performed reliably by an expert on the application of the components*, one who need not be an expert on the internals of the component itself. In this way, the component performs a role similar to that played by the architecture of a processor, which "orthogonalizes" the software developer from the underlying hardware implementation. In a component-based approach, the systems programmer plays the role of application developer. By enabling the person who understands the "market" for the system under development, it is possible to rapidly configure and iterate applications to suit the needs of particular situation.

Recall that at the system level it is crucial to distinguish between the *functionality* to be implemented and the *architecture* (software and/or hardware) and *microarchitecture* upon which it is to be implemented. In a component-based approach to design, the most critical aspect of the implementation architecture is not the choice of the components themselves, but rather the *communication infrastructure*—the protocols and interfaces—used to compose the components into a system that implements the desired functionality. So while we advocate a component-based approach to design reuse, we also believe future research emphasis must be placed on the formalisms (both combinational and sequential) and implementation details associated with the composition of multiple components. Since chips are getting bigger and faster, and since a single clocked-synchronous approach to managing on-chip concurrency will be wasteful of power or performance (or both!), we believe that a major research challenge in the design of such an infrastructure is the efficient and reliable implementation of concurrency, including the interfaces to the external world. The optimal implementation and verification of (possibly concurrent) reliable communication among a collection of components on a complex chip and their associated chip-level input–output (I/O) requirements will be a major research challenge in the years ahead. Here, optimality is measured in terms of some combination of performance, power, and cost, and the interpretation of "reliable" will be situation-dependent. For example, some values may need to be transferred between components within a specifically bounded interval, while others may have much less stringent requirements for reliable communication. Hence, we do not believe that a single, general approach (e.g., a scheme implemented on a single, standard bus) is a viable overall solution. Rather, we must develop a set of *general principles* that can be applied to and adapted for many different communication requirements and implementation strategies. These principles must be couched in a formal framework that allows for formal, or complete, verification of the critical properties of the communication medium, when such verification is necessary to guarantee reliable and correct functionality. (For example, while the component-based approach is clearly very interesting for system design in general, it is important to emphasize that it is semiformal and relies on implicit assumptions about the models of concurrency of the components.)

The situations in which component-based approaches have failed in the past are where the interfaces and protocols are complex and where components are developed by different groups in different locations. Usually, it is the verification issue that limits the utility of such systems; this represents a major research challenge, as we have discussed in Section III-B.

B. Platform-Based Design

Over the past 15 years, as chips have become more complex we have continued to raise the level of abstraction at which the design is described—from transistor-level descriptions, to gate-level schematics, to the register-transfer level and even to domain-specific behavioral descriptions today—to keep up with growing design complexities. However, the level at which the design is handed off to the manufacturer has not really evolved beyond a gate-level netlist. As a result, we currently rely on the synthesis process to reliably predict the eventual performance of a design at the netlist level from a very high level of design entry. As discussed in Section III-C, the increasing role of physical embedding effects, especially interconnect-related effects, has made such predictions virtually impossible. Just as the design handoff level was raised from mask-level rectangles, to transistors, and eventually to logic gates in the early 1980s with the advent of the ASIC era, we believe that it is essential to *raise the handoff level* again. Given a predictable path to implementation from a higher level of abstraction than the gate level, we believe designers will again achieve the short design times and predictable implementations they seek.

It appears that the most reasonable level to propose for such a handoff is the boundary between architecture and microarchitecture, which we refer to as the architectural level of representation in Fig. 5. By using a component-based approach where the components are the major elements of a microarchitecture and by applying a communication-based integration approach, its should be possible to significantly improve the predictability and the robustness of implementation from much higher levels than we are accustomed to today. Of course, this will also require major advances in the logic/physical implementation phases of design, where a reliable and predictable implementation from the microarchitectural level is required. Possible approaches that lead to such advances are presented in Section IV-D below.

We must also consider the "bounded" nature of the handoff. A primary reason for the success of the ASIC methodology, and for why gate-level handoff has been the industry standard for so many years, is that the number

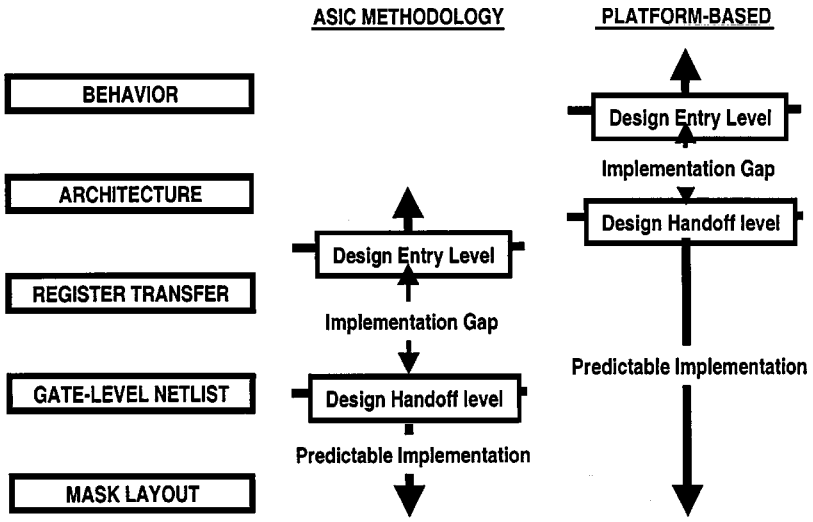

Fig. 5. Design entry and design handoff levels for ASIC and platform-based approaches.

of components the designer can choose from at the gate level is finite. That is, the library can be precharacterized, with the characterized models used directly for prediction of function and performance.[1] For an architectural-level handoff to work as effectively, it is necessary to provide a similar form of limitation to the classes of designs that can be implemented from the microarchitectural level. Such an approach is also important for the implementation of significant component-level reuse as well. A family of architectures that allows substantial reuse of microarchitectural components is what we refer to as a *hardware platform*. We believe that hardware platforms will take the lion's share of the IC market in the years ahead. However, the concept of hardware platform by itself is not enough to achieve the level of application software reuse that we seek. To be useful, the hardware platform must be abstracted such that the application software sees a high-level interface to the hardware, which we call the *application program interface* (API). There is a software layer, or *software platform*, that is used to perform this abstraction: the software platform wraps the different parts of the hardware platform, i.e., the programmable cores and the memory subsystem via a real-time operating system (RTOS), the I/O subsystem via the device drivers, and the network connection via the network communication subsystem. We refer to the combination of the hardware and the software platforms as a *system platform*. We believe that platform-based approaches to the implementation of microelectronic systems—where platforms may be implemented as collections of hardware components and their interconnects, as collections of software components and an underlying execution model, or as combinations of the two—represent a major methodology change for the semiconductor industry and are likely to comprise the most effective approach to the implementation of design reuse.

[1]In today's ASIC methodology, the specific interconnections used in a design cannot be precharacterized in the same way, leading to the timing closure problem described earlier.

Last, recall that economics of NRE and time-to-market dictate a paradigm of reuse. The mapping of function to architecture is an essential step in moving from conception to implementation. But when mapping the functionality of the system to an integrated circuit, the economics of chip design and manufacturing determine the quality and the cost of the system. It is critical to find common architectures that can support a variety of applications and yet reduce design costs through reuse. In particular, since system designers will more frequently use software to implement their products, there is a need for design methodologies that allow the substantial reuse of software. This implies that the basic architecture of the implementation is essentially "fixed," i.e., the principal components should remain the same within a certain degree of parameterization. For embedded systems, which we believe are going to be the dominant share of the electronics market, the "basic" architecture consists of programmable cores, I/O subsystem, and memories.

C. Programmable Embedded Systems

The trends of increasing NRE cost, time-to-market pressure, verification cost, and complexity of contexts in which designs are used (making *in situ* debugging almost essential) together lead to increased value of programmable solutions to today's designs. Coupled with the concept of a precharacterized, optimized, and preverified platform (i.e., microarchitectural family), we obtain *programmable platforms* as a likely future design methodology. Here, *programmable* encompasses the full range of "commit-after-fabrication" devices—not only Von Neumann-style instruction-set processors, but also parameterizable application-specific accelerators, and configurable processing modules that employ spatial programming in the style of field-programmable gate arrays (FPGAs). Such a wide range of options is necessary to meet the stringent energy, performance, reliability, and cost requirements imposed by the embedded applications where most of these SOCs will be deployed. We foresee *parameterized "standard programmable platforms"* for the imple-

mentation of embedded systems, replacing the "unique assemblies of components" approach currently taken for each new design. While the idea of a platform itself is not new (e.g., Intel ×86 architecture, or Texas Instruments TMS 320 DSP family), a broad movement toward platform-based design would represent a major discontinuity for the semiconductor industry.

For the programmable platform-based design approach to be viable, the critical challenge is understanding what it means to program a complex SOC efficiently, i.e., "What is the programmer's model?" or "How should the programmer view the underlying hardware and I/O systems?" On one hand, we want to hide as many of the details of the underlying implementation as possible, but on the other hand we want to make visible a sufficient level of control that the application programmer can develop an efficient solution—in terms of performance, power, and reliable functionality. From the discussion in Section III-A, any successful solution to this problem must also comprehend the issue of concurrency, both in terms of instruction-level parallelism as well as more general forms of concurrency. This is particularly important when viewed in the context of power consumed per unit of work done [e.g., milliwatts per million instructions per second (milliwatts/MIPS)]. Today, depending on the application domain and nature of the programmable processor, programmed implementations of many common algorithms can be as much as 10000 times worse than a hardware-only single-chip solution in terms of this metric, an unacceptable penalty for many applications. Of course, this is not a new problem in the general sense. The design of efficient, instruction-level parallel processor architectures, microarchitectures, and associated compilation technology is a very active field of research today. However, much of this work has been targeted toward high-end processors rather than embedded systems, and the issue of optimal power/performance has not been addressed comprehensively.

The major aspects of this problem are illustrated above in Fig. 6. To gain the maximum benefit from the final silicon, we believe any successful research agenda must break down the conventional hardware/software boundary and examine more thoroughly the possible advantages of simultaneously exploring architectural and microarchitectural tradeoffs in conjunction with programming models, software, and compilation. Due to our need to deal with concurrency explicitly and efficiently, this work must also transcend aspects of the conventional operating system boundary.

D. Reliable and Efficient Physical Design

With the necessary functional partitioning for system-level design, it is apparent that the initial design of any gigascale IC must be hierarchical and block-like. Ideally, this would not require that the physical instantiation of this design conform to rigid fixed-shape blocks, nor would it require that hierarchical block-based design styles restrict the ability to perform logic redesign and resizing within blocks during physical assembly of blocks. Once the blocks of a gigascale hierarchical design are combined for physical design, a

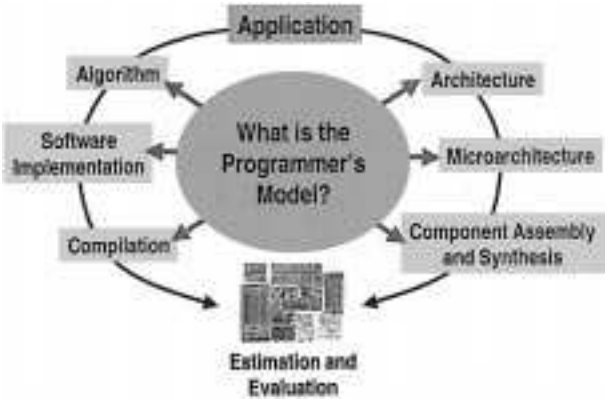

Fig. 6. Major aspects of a programmable solution.

completely flattened physical design that includes resizing and remapping could *theoretically* produce superior delays, hence more readily achieve timing closure, due to the increased degrees of freedom in optimizing the design. Simply put, a rigid block-based physical design with buffered global wiring among the blocks has only a small subspace of the flattened design's solution space. However, how much of that space we can effectively explore, and how thoroughly we can explore it, is an important open question.

1) Constructive Blocks: If the gates could also migrate among blocks, and the blocks could change in shape, it is apparent that the a much larger portion of the physical design space could be explored. Importantly, as we approach gigascale integration, if the block sizes decrease relative to the chip size as suggested in [2], one could consider the redesign and/or remapping of a block as analogous to a single gate resizing as it is performed today. In contrast to gate sizing, to simply resize the gates along the hierarchical periphery is obviously suboptimal in terms of timing. For example, constant-delay-effort synthesis rules for even the simplest of examples would be violated in such cases. But just as with gate resizing, block redesigning is best performed with physical design information, especially for future DSM technologies. Therefore, one of the most important challenges that lies ahead is to develop new methodologies for on-demand remapping and redesign of blocks.

2) Measurement and Prediction: Even with a constructive block capability, some form of performance prediction is required prior to physical design. Just as we today have some notion of the capabilities of a gate, cell, or small block over various sizes, a similar capability would be required for predicting the performance of *soft*, constructive blocks, prior to their actual design. Beyond obvious statistical methods for prediction (i.e., estimates of tool outcomes based on parameters of design artifacts and design optimization heuristics), design technology can exploit a variety of methods for building *predictability* into design tools and methodology. As one example, circuit design styles (based on programmable logic arrays (PLAs), or power-signal-signal-ground (PSSG) style-shielded interconnect design, etc.) can improve timing predictability by enforcing fixed stage delays or by limiting the amount of noise coupling. As a second example, any lack of "good

predictions" in forward synthesis can be made up for by "enforceable assumptions." The constant-delay methodology referred to above is such an "enforceable assumption": one assumes that the delay of a given gate is independent of its fanout and load (thus simplifying RTL synthesis, technology mapping, and other tasks), then makes the assumption true via constraints on downstream sizing, buffering and clustering operations. A third example approach is to adopt design methodologies that remove some of the requirements for predictability. For example, we have noted that the number of clock cycles required for two components to communicate may depend heavily on the physical placement of these components. Globally asynchronous locally synchronous design styles can eliminate any need for global clocking, and will in any case be required by the 50-nm process generation [60]. Latency-independent synchronization can apply protocols to guarantee that the design behaves correctly no matter how many clock cycles separate components.

3) Design Reuse at the Physical Layout Level: With shrinking design resources and tighter time-to-market schedules, reuse of intellectual property (IP) plays an enormous role in gigascale design. The increasing number of fabless design houses and use of multiple fabs makes reuse most practical at higher levels of design abstraction. For example, reuse of soft and firm IP is clearly a superior design model as compared to hard IP, given the way in which manufacturing is evolving. The ability to construct large blocks from soft IP would greatly enhance design reuse and gigascale design in general.

E. New Approaches to System-Level Test

While the vast heterogeneity and complexity of the components may make one unified testing methodology difficult to achieve, it is obviously advantageous to develop component-level and system-level techniques to make the embedded components, and the systems comprised of the components, self-testable. The self-test mechanisms should be able to respond to configurability and customizability of the components, with the ability to synthesize the self-tests automatically. Furthermore, it is essential to develop techniques to make the SOC self-diagnosable, facilitating both manufacturing testing and on-line testing, utilizing the self-testability features of the system. In utilizing the configurability, modularity, and customizability available in a SOC, the ultimate goal would be to develop self-repair techniques for safety-critical applications, based on the on-chip self-testing and self-diagnosis capabilities inserted.

1) Component-Level Self-Test Methodologies: While self-testability can be achieved for components with regular structures, such as memory and DSP components, further investigation is necessary for new ways of making control-intensive components (interface and communications components, and programmable components like processor and micro-controller cores) self-testable.

Almost all SOCs consist of some programmable components such as processor cores, DSP cores, or FPGA cores. Such programmable components can be used for test objectives such as test generation and response analysis. For such programmable components, new self-test techniques that can obviate the need for incorporating special BIST hardware [e.g., linear feedback shift registers (LFSRs)] in the component are being investigated in the research community today [91]–[93]. It is possible to use the instruction-set of the component to synthesize self-test programs that, when executed, will result in high stuck-at and delay fault coverages. For example, the registers of the processor can be loaded with random patterns by executing random number generation programs [93], instead of having to add special hardware. However, not all parts of the processor will either preserve the randomness or be amenable to random patterns. For such a part, e.g., the instruction decode unit, synthesizing a random sequence of instructions that can provide a high fault coverage for the unit is desirable. It may be necessary to add special test instructions that may be added to the instruction set and used in the self-test program to enhance the fault coverage of the self-test program. The test instructions will be particularly helpful in testing the complex controllers of a processor core, by generating state transitions that are not possible with regular instructions. The test instructions will also have the ability to introduce "soft" test points, to enhance the random-pattern testability of the processor core.

2) System-Level Test, Test Access Interface, and Diagnosis: For embedded systems consisting of programmable components, we believe that a promising direction lies in the investigation of a new self-testing methodology for the entire system, using the instruction-set of the processor core to synthesize test programs for each existing component in the SOC.

While a self-test for the processor core can be generated using the component-level analysis and synthesis approach described above, generation of the test programs for the other components will be achieved by using self-testability requirements that need to be specified for each component in the component's test protocol. Subsequently, the processor core can test a component by executing the assigned test program that will exercise the component-under-test and reading and analyzing the component's response. Delivery mechanisms of the test programs to the other components may not be trivial. Modifications to the integration and communication architectures and protocols used in the system-on-chip, like the bus arbiter, may be necessary.

Based on the ability to test each component, techniques for self-diagnosis, wherein a self-diagnosis program will be generated to identify the location of an error in the SOC, need to be further developed. Using the self-testing and self-diagnosis capabilities inserted in a SOC, on-line testing and debugging capabilities for system maintainability then become feasible. For safety-critical applications, self-repair techniques exploiting the configurability, and customizability available in the SOC design paradigm should be further investigated. A key issue for investigation is how the reconfigurability allowed by adding coprocessors, special instruction sets, and peripheral units, to a modern embedded processor core can be utilized for self-repair.

3) Fault Modeling, Test Generation, and Diagnosis for DSM Defects: Noise faults must be detected during both design verification and manufacturing testing. When coupled with process variations, noise effects can produce worst case design corners—combinations of design characteristics that represent extreme operating conditions. These corners must be identified and checked as part of design validation. This task is extremely difficult, however, because noise effects are highly sensitive to the input pattern and to timing. Timing analysis that cannot consider how noise effects influence propagation delays will not provide an accurate estimate of performance nor will it reliably identify problem areas in the design.

An efficient test generation method must be able to generate validation vectors that can excite these corners. To do this, it must integrate accurate timing information when the test vectors are derived. For manufacturing testing, existing test generation techniques must be augmented and adapted to new failure conditions introduced by nanometer technology. Tests for conventional fault models, such as stuck-at and delay, obviously cannot detect these conditions. Thus, to check worst case design corners, test vectors must sensitize the faults and propagate their effects to the primary outputs, as well as produce worst case noise effects [87]. They must also scale to increasingly larger designs.

a) Power supply noise: For a highly integrated system-on-a-chip, more devices are switching simultaneously, which increases power supply noise. One component of this noise, inductive noise, results from sudden current changes on either the package lead or wire/substrate inductance. The other component, net *IR* voltage drop, is caused by current flowing through the resistive power and ground lines. The noise can cause a voltage glitch on these lines, resulting in timing and/or logic errors. Moreover, large current densities through the power supply lines can cause electromigration, which in turn can cause short or open circuits. To activate these defects and propagate them to the primary outputs, test vectors must be carefully selected.

Power supply noise can affect both reliability and performance. It reduces the actual voltage level that reaches a device, which in turn, can increase cell and interconnection propagation delays. SPICE simulations show that a 10%–15% voltage drop during cell transition can cause a 20%–30% increase in cell propagation delay. If the cell is a clock buffer or is in the timing-critical path, this delay deviation could cause serious clock-skew problems or a nontrivial increase in the critical path delay.

One way to detect these effects is to apply delay tests. Unfortunately, most existing delay techniques are based on simplified, logic-level models and cannot be directly used to model and test timing defects in high-speed designs that use DSM technologies. New delay testing strategies are needed to close the gap between the logic-level delay fault models and physical defects. The tests must produce the worst case power supply noise along the sensitized paths and therefore, cause the worst case propagation delays on these paths [86], [90].

b) Crosstalk effects: The increased design density in DSM designs leads to more significant interference between the signals because of capacitive coupling, or crosstalk. Crosstalk can induce both Boolean errors and delay faults. Therefore, ATPG for worst case crosstalk effects must produce vectors that can create and propagate crosstalk pulses as well as crosstalk-induced delays. Crosstalk-induced pulses are likely to cause errors on hazard sensitive lines such as inputs to dynamic gates, clock, set/reset, and data inputs to flip-flops. Crosstalk pulses can result in logic errors or degraded voltage levels, which increase propagation delays. ATPG for worst case crosstalk pulse aims to generate a pulse of maximum amplitude and width at the fault site and propagate its effects to primary outputs with minimal attenuation [3].

Studies show that increased coupling effects between signals can cause signal delay to increase (slowdown) or decrease (speedup) significantly. Both conditions can cause errors. Signal slowdown can cause delay faults if a transition is propagated along paths with small slacks. Signal speedup can cause race conditions if transitions are propagated along short paths. To guarantee design performance, ATPG techniques must consider how worst case crosstalk affects propagation delays.

Finally, fault diagnostic methods need to be enhanced for DSM faults. Existing fault diagnosis techniques are based on the stuck-at fault model, which is two generations removed from the DSM fault models and will not be effective in DSM devices. Increasing pressure to shorten the process debugging time and cost exacerbates this issue. Such techniques must be dynamic, instead of static, to diagnose DSM faults. The diagnostic method need to be compatible with the emerging BIST methodology as well. One key diagnostic focus should be on ac failures due to the fact that most of the new defect types of DSM devices are more likely related to timing and noise-related faults.

4) Supporting Heterogeneity: Test Technologies for Analog/RF Circuits: For most mixed-signal high-end ICs, the analog circuitry accounts for only a small fraction of the total silicon area while a major fraction of the test equipment investment and test time is devoted to the analog parts. Therefore, analog BIST, which could significantly reduce the need of expensive external analog testers, shorten the test time, and minimize the measurement noise, offers great promises. BIST for analog blocks requires integrating analog signal sources and measuring circuitry on chip. Moreover, these circuits will be used to characterize others and therefore, they should involve little or no calibration and occupy minimal area.

A promising approach to analog self-test is to use DSP-based techniques for both stimulus generation and response analysis. DSP-based testing techniques utilize digital techniques for signal generation and for response analysis. For SOCs with on-chip digital-to-analog and analog-to-digital converters and DSP cores, such components can be utilized as part of the signal generator and the response analyzer. This direction has obvious advantages such as lower hardware overhead, lower performance intrusion, and

higher flexibility. Not only is most of the test logic coming from functional logic, but the same test hardware can be used to perform a multitude of test functions.

Stimulus generation using delta-sigma modulation is also an attractive alternative self-test approach [94]. The hardware requirement includes memory storage for storing a 1-b digital sequence (i.e., a sequence of binary numbers), a 1-b DA converter, and a low-pass filter. Due to its oversampling nature, this technique has high tolerance to the required DA converter and filter. For response analysis, a couple of approaches are being investigated. The first approach is to utilize a 1-b delta-sigma encoder (without the output decimator and filter) that is either part of the original design or dedicated BIST structure to encode the signal (dc or ac) of interest as a 1-b digital stream (i.e., a stream of binary numbers) [88]. The advantages are: 1) reduced signal processing power, e.g., the multiplication operation needed for DFT analysis can be realized by addition and multiplication and 2) the desired dc measurement accuracy can be attained by adjusting the test time. The second approach is to use a linear ramp signal together with an analog comparator for measuring the difference between two analog quantities [89]. The common offset voltage of the analog comparator has little effect on the test accuracy as it is canceled out in the "difference" operation.

a) Potential extension to RF testing: Through the application of heterodyning or undersampling, the DSP-based self-test schemes can further extend their frequency measurement capabilities to the IF or RF frequency bands. The basic idea is to translate the IF/RF response back down into the baseband range of the digitizer for DSP processing. By doing so, the DSP-based self test structure could be maintained for RF measurements. The issues of minimizing the area of the test circuitry, minimizing/eliminating performance intrusion, maximizing the test quality, and characterizing the noise effects need to be carefully investigated.

F. Structure of the Design-Manufacturing Interface

Recall from above the assertion that the "cost contradiction" inherent in the ITRS compels improvement of manufacturing efficiency, including the practice of *design for manufacturability* (DFM) at all levels of design abstraction. We believe that associated structural changes at the design-manufacturability interface will arise as follows. First, it is likely that small design houses will have a tendency to focus on higher levels of design abstraction, e.g., some will focus on fast "IP assembly" to avoid the challenges of DSM physical reality, and all are likely to seek market advantage in domains located farther and farther from physical reality of the design implementation. On the other hand, the key mission of silicon providers will be high manufacturing efficiency, i.e., *full utilization* of available manufacturing hardware will be their only relevant goal. These two trends are likely to result in a widening gap between the design and manufacturing domains, which will constitute an opportunity for new kinds of design specialization/services. Some of these are already emerging, e.g., services related to characterization and test of IP blocks are already offered by key silicon providers.

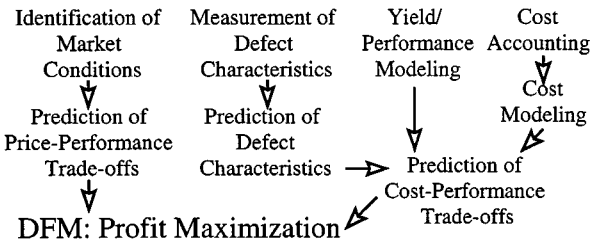

Fig. 7. Global view on design for manufacturability (DFM) and related issues.

Ultimately, however, interface needs between design and manufacturing are likely to be served by independent "IP bank," "IP design/redesign," and "test and yield ramp" services that have yet to emerge. Each of these domains must evolve from existing domains of IC design expertise via investments in such areas as: 1) design for manufacturability (DFM), broadly defined as a domain allowing for profit maximization [97]–[100] (see Fig. 7); 2) performance and cost forecasting, which will be used to assess economic aspects of design decisions at all levels of design abstraction [101]–[105]; and 3) design and test for defect observability, which is needed for predesign assessment of true capabilities of potential silicon provider partnerships and for adequate test development purposes [106]–[108]. From the research perspective, broad-based and collaborative attention from a range of disciplines (and with respect to a number of "less-scientific" considerations) will be required in the DFM domain.

G. Measurement and Continuous Improvement of Design Technology

Earlier discussion has described how various factors cause today's design processes to be less able to meet project goals: more designs miss time-to-market windows and/or end up substantially over budget. This trend is accelerated as designers rely on IP reuse and integration to meet turnaround time requirements, since the complexity of the design process is increased along such axes as cost, testability, etc. The result is that today's design processes are more "art" than "science": temporary solutions, unique to individual projects, are created based on the intuition of senior engineers. Such solutions typically last for one project only, while the basic issue of unpredictable design success remains unaddressed. In this light, a fundamental new avenue of research concerns the understanding, diagnosis, optimization, and prediction of the system design process itself.

Design process optimization refers to the continuous optimization of a design process. In automobile, steel, or semiconductor manufacturing, process optimization is a well-established precept. Furthermore, before a process can be optimized on a continuous basis, it must first be measured. Today there are no standards or infrastructure for measuring and recording the semiconductor design process. As a result, a product team cannot quantify inefficiencies in its design process, and subjective opinions are formulated as to why a

given project failed or succeeded (failure may be generically blamed on "CAD tools" or "inexperienced design team," while real causes of failure remain unidentified).

Two basic gaps prevent us from measuring the design process. First, the data to be measured are not available. Second, and more important, we do not always know what data should be measured. Some metrics of tool performance or design artifacts are "obviously useful," e.g., number of placeable objects, number of signal nets left unrouted after detailed routing, maximum negative slack over all timing paths, etc. Other metrics are less obviously useful, e.g., it is not clear whether the number of literals after logic optimization has any relationship to the quality of the resulting netlist from a physical implementation perspective. Finally, yet other metrics are almost impossible to discern *a priori*, e.g., it may turn out that the number of clocks in the design, or the number of specification changes, might be the best predictor of project success.

The research and infrastructure changes that will bridge these two gaps must be developed along two main fronts. First, relevant and useful metrics of the design process, of design artifacts, of particular tool classes, etc., must be identified, and inefficiencies in the design process must be diagnosed. To this end, techniques from continuous process improvement (CPI)—a methodology that analyzes a (design) process and optimizes it on a continuous basis—will likely be applicable. CPI is the best known embodiment of the "measure, then improve" precept and is currently applied in most manufacturing industries. When many parameters are available to be measured and improved, they cannot all be improved at once (e.g., total chip design effort trades off against chip area/power/speed). Techniques for identifying improvements due to CPI are given by, e.g., [45]. Reference [33] gives a method for identifying the most important metrics for collection, to reduce the complexity of data gathering. Data mining [32], [36], [39] (the AI- and statistics-based extraction of predictive information from large databases) and visualization [27], [34], [37] technologies will also enhance human understanding of correlations and trends within continually evolving design processes.

Second, a standard infrastructure (semantics, schema, collection infrastructure, etc.) for design process metrics must be established. An example of infrastructure for project tracking is found in the N-dim design support system [29], [38], [40], which collects and maintains design information, and also analyzes this information for project management purposes. N-dim tracks changes so that design information is always relevant and complete; information storage is transparent to possible changes in the underlying design process, so that no "loose data" is lost in translation to new projects. The N-dim system is tightly focused on project data tracking and does not address the improvement of the design process, hence it complements the first research front described above. Rapid growth in Internet and e-commerce infrastructure will also aid in the creation of infrastructure for metrics, since many of the technical challenges appear similar. Such infrastructure has been discussed mostly in the context of distributed and collaborative design. Distributed web-based design environments include Fujitsu's IPSymphony [31], the Berkeley WELD project [44], and the VELA project [43]. E-commerce efforts toward time- or token-based tool use, possibly on external server farms (application service providers), include [30], [35], [41]. Such web and server farm infrastructure is highly conducive to recording of design process metrics.

V. Summary

Without design technology, it would be impossible to implement, verify, and test the complex, single-chip electronic systems that are the foundation of today's information technology revolution. It is through design technology that the ideas and objectives of the electronic systems designer are transformed into reality, and the quality of the design tools and associated methodologies determine the design time, performance, cost, and correctness of the final system product. While it would be very useful to know how close a given piece of design technology is to its "*fundamental limits*" of performance—e.g., as a synthesis, verification, or test system—we observe that such a question is ill-posed and intractable: in some sense, there are only "*fundamental limitations*" of design technology. These limitations include: 1) the multiobjective nature of the design problem, which inherently casts design as a set of tradeoffs; 2) the external context for design, which partially decouples "optimal use of silicon" from the underlying design technology; and 3) the (intractable) optimization-centric nature of design, which forces heuristic approaches throughout all aspects of design. This last is perhaps the sole "fundamental limit" in design technology.

Our discussion has summarized the mainstream methodologies used by CMOS silicon designers today and—against the backdrop of International Technology Roadmap for Semiconductors (ITRS) forecasts—pointed out fundamental limitations to the ability to achieve "optimal" design quality within reasonable resources. In key areas of today's mainstream design flow, we have identified factors that limit progress or pointed out the work that must be done to obtain such an understanding. We have also presented a number of potential solutions to these problems, in the form of methodological approaches and major outstanding research questions that are being considered actively within the design technology research community.

Finally, many aspects of design technology have been beyond the scope of our discussion due to space limitations. Among these, we would highlight the problem of collaborative design, the development of distributed design tools and methodologies, the role of human factors and integration factors in design technology, and the special requirements of domain-specific design technology (e.g., for analog and RF subsystems).

Acknowledgment

This paper reflects a number of shared visions within the SIA/DARPA MARCO Gigascale Silicon Research Center (GSRC) for Design and Test. The MARCO GSRC is an

experiment in the area of university-based collaborative research. It involves close collaboration among faculty, with industry, and with other Department of Defense-sponsored research groups. With a stated 8–12 year research agenda, the GSRC has evolved and refined its research emphasis over the past year; the authors believe they have made significant progress toward identifying and formulating key methodological and technical research problems that are at the heart of design technology for integrated systems in the years to come. Just as important, every member of the research team has a shared understanding of this comprehensive research vision. The authors would like to thank MARCO, DARPA, and their respective constituents for supporting the development of the unique research community that makes up the GSRC. The faculty currently working within the GSRC are listed below. The ideas presented above are the result of many hours of collaborative discussion within the GSRC: all of the investigators listed below, as well as the many students, post-doctoral researchers, and industrial collaborators working within the GSRC, share the credit for the ideas presented in this paper: J. Babb, Princeton University, R. Brayton, University of California (UC) at Berkeley, R. Bryant, Carnegie-Mellon University, R. Carley, Carnegie-Mellon University, T. Cheng, UC Santa Barbara, E. Clarke, Carnegie-Mellon University, J. Cong, UC Los Angeles, W. Dai, UC Santa Cruz, G. De Micheli, Stanford University, S. Dey, UC San Diego, D. Dill, Stanford University, T. Henzinger, UC Berkeley, A. B. Kahng, UC San Diego, K. Keutzer, UC Berkeley, E. Lee, UC Berkeley, S. Malik, Princeton University, W. Maly, Carnegie-Mellon University, R. Newton, UC Berkeley, L. Pileggi, Carnegie-Mellon University, J. Rabaey, UC Berkeley, K. Roy, Purdue University, M. Marek-Sadowska, UC Santa Barbara, K. Sakallah, University of Michigan, A. Sangiovanni-Vincentelli, UC Berkeley, A. Strojwas, Carnegie-Mellon University.

REFERENCES

[1] S. Hojat and P. Villarrubia, "An integrated placement and synthesis approach for timing closure of powerPC microprocessors," in *Proc. Int. Conf. Computer Design*, 1997, pp. 206–210.

[2] D. Sylvester and K. Keutzer, "Getting to the bottom of deep submicron," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1998, pp. 203–211.

[3] J. Grodstein, E. Lehman, H. Harkness, B. Grundmann, and Y. Watanabe, "A delay model for logic synthesis of continuously sized networks," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1995, pp. 458–462.

[4] I. Sutherland and R. Sproull, "The theory of logical effort: Designing for speed on the back of an envelope," in *Proc. Conf. Advanced Research VLSI*, 1991.

[5] R. H. J. M. Otten, "Global wires harmful," in *Proc. ACM Int. Symp. Physical Design*, 1998, pp. 104–109.

[6] S. Sengupta *et al.*, "Defect-based test: A key enabler for successful migration to structural test," *Intel Tech. J.*, First quarter 1999.

[7] National Science Foundation Workshop on Future Research Directions in Testing of Electronic Circuits and Systems (1998). [Online]. Available: http://yellowstone.ece.ucsb.edu/NSF_WORKSHOP/

[8] R. Kapur and T. W. Williams, "Tough challenges as design and test go nanometer," *IEEE Computer*, vol. 32, no. 11, pp. 42–45, 1999.

[9] Microelectronics Advanced Research Corporation (MARCO) [Online]. Available: http://marco.fcrp.org/

[10] Defense Advanced Research Projects Agency (DARPA) [Online]. Available: (DARPA) http://www.darpa.mil

[11] International Technology Roadmap for Semiconductors [Online]. Available: (ITRS) http://www.itrs.net/ntrs/publntrs.nsf

[12] Gigascale Silicon Research Center (GSRC) [Online]. Available: http://www.gigascale.org

[13] W.-J. Sun and C. Sechen, "Efficient and effective placement for very large circuits," *IEEE Trans. Computer-Aided Design*, vol. 14, no. 3, pp. 349–359, 1995.

[14] W. Swartz and C. Sechen, "Timing driven placement for large standard cell circuits," in *Proc. ACM/IEEE Design Automation Conf.*, 1995, pp. 211–215.

[15] H. Eisenmann and F. M. Johannes, "Generic global placement and floorplanning," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1998, pp. 269–274.

[16] H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly, and L. Todd, *Surviving the SOC Revolution: A Guide to Platform Based Design*. Norwell, MA: Kluwer, 1999.

[17] G. Martin and B. Salefski, "Methodology and technology for design of communications and multimedia products via system-level IP integration," in *Proc. Design Automation Test Eur.*, 1998.

[18] J. R. Hauser and J. Wawrzynek, "Garp: A MIPS processor with a reconfigurable coprocessor," in *Proc. IEEE Symp. Field Programmable Gate Arrays Custom Computing Machines*, 1997, pp. 12–21.

[19] F. Balarin, M. Chiodo, P. Giusto, and H. Hsieh, *Hardware–Software Co-Design of Embedded Systems: The Polis Approach*. Norwell, MA: Kluwer, 1997.

[20] J. L. Pino, S. Ha, E. A. Lee, and J. T. Buck, "Software synthesis for DSP using Ptolemy," *J. VLSI Signal Processing*, vol. 9, no. 1–2, pp. 7–21, 1995.

[21] G. Berry. (1998) The foundations of Esterel. [Online]. Available: http://www.inria.fr/meije/Esterel

[22] *The Java Programming Language*, 1997.

[23] K. Arnold, B. O'Sullivan, B. W. Scheifler, J. Waldo, A. Wollrath, and B. O'Sullivan, *The JINI Specification*. Reading, MA: Addison-Wesley, 1999.

[24] C. G. Bell and A. Newell, *Computer Structures: Readings and Examples*. New York: McGraw-Hill, 1971.

[25] J. Hennessey and D. Patterson, *Computer Architecture: A Quantitative Approach*. New York: Morgan Kaufmann, 1990.

[26] P. S. Adler and K. B. Clark, "Behind the learning curve: A sketch of the learning process," *Manage. Sci.*, vol. 37, no. 3, pp. 267–281, 1991.

[27] S. S. Bhowmick, S. K. Madria, W.-K. Ng, and E. P. Lim, "Data visualization in a web warehouse," in *Proc. Advances Database Technologies*, 1999, pp. 68–80.

[28] J. S. Busby, "Effective practices in design transfer," *Res. Eng. Design*, vol. 10, no. 3, pp. 178–188, 1998.

[29] [Online]. Available: http://www.ndim.edrc.cmu.edu

[30] [Online]. Available: http://www.EDAmall.com

[31] B. Fuller, "Fujitsu shows browser-based design methodology," *EE Times*, Apr. 1999.

[32] M. L. Gargano and B. G. Raggad, "Data mining—A powerful information creating tool," *OCLC Syst. Services*, vol. 15, no. 2, pp. 81–90, 1999.

[33] A.-P. Hameri and J. Nihtila, "Computerized product process: Measurement and continuous improvement," *Res. Eng. Design*, vol. 10, no. 3, pp. 166–177, 1998.

[34] M. Jern, "Information drill-down using web tools," in *Proc. Visualization Scientific Computing*, 1997, pp. 9–20.

[35] C. Matsumoto, "Web-based tools gathering momentum," *EE Times*, June 1999.

[36] C. J. Meneses and G. G. Grinstein, "Categorization and evaluation of data mining techniques," in *Proc. Int. Conf. Data Mining*, 1998, pp. 53–80.

[37] S. O. Rezende, R. B. T. Oliveira, L. C. M. Felix, and C. A. J. Rocha, "Visualization for knowledge discovery in database," in *Proc. Int. Conf. Data Mining*, 1998, pp. 81–95.

[38] J. L. Robertson, E. Subrahmanian, M. E. Thomas, and A. W. Westerberg. Management of the design process: The impact of information modeling. [Online]. Available: http://www.ndim.edrc.cmu.edu/papers/manodespro.pd

[39] M. S. Sousa, M. L. Q. Mattoso, and N. F. F. Ebecken, "Data mining: A database perspective," in *Proc. Int. Conf. Data Mining*, 1998, pp. 413–431.

[40] E. Subrahmanian, Y. Reich, S. L. Konda, A. Dutoit, D. Cunningham, R. Patrick, M. E. Thomas, and A. W. Westerberg, "The *N*-Dim approach to creating design support systems," in *Proc. ASME Design Technical Conf.*, 1997.
[41] "Synopsys' FPGA Express now available on the Internet from Toolwire, Inc.," *Business Wire*, Feb. 2000.
[42] D. M. Upton and B. Kim, "Alternative methods of learning and process improvement in manufacturing," *J. Oper. Manage.*, vol. 16, pp. 1–20, 1998.
[43] [Online]. Available: http://www.cbl.ncsu.edu/vela
[44] [Online]. Available: http://www-cad.eecs.berkeley.edu:80/Respep/Research/weld
[45] W. I. Zangwill and P. B. Kantor, "Toward a theory of continuous improvement and the learning curve," *Manage. Sci.*, vol. 44, no. 7, pp. 910–920, 1998.
[46] J. Alpert, A. E. Caldwell, T. F. Chan, D. J.-H. Huang, A. B. Kahng, I. L. Markov, and M. S. Moroz, "Analytic engines are unnecessary in top–down partitioning-based placement," *VLSI Design*, vol. 10, no. 1, pp. 99–116, 1999.
[47] A. E. Caldwell, A. B. Kahng, and I. Markov, "Can recursive bisection alone produce routable placements," in *Proc. IEEE Design Automation Conf.*, 2000, pp. 477–482.
[48] C.-C. Chang, J. Lee, M. Stabenfeldt, and R.-S. Tsay, "A practical all-path timing-driven place and route design system," in *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, 1994, pp. 560–563.
[49] A. H. Chao, E. M. Nequist, and T. D. Vuong, "Direct solutions of performance constraints during placement," in *Proc. IEEE Custom IC Conf.*, 1990, pp. 27.2.1–27.2.4.
[50] J. Frankle, "Iterative and adaptive slack allocation for performance-driven layout and FPGA routing," in *Proc. IEEE Design Automation Conf.*, 1992, pp. 539–542.
[51] W. Gosti, A. Narayan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Wireplanning in logic synthesis," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1998, pp. 26–33.
[52] P. S. Hauge, R. Nair, and E. J. Yoffa, "Circuit placement for predictable performance," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1987, pp. 88–91.
[53] Y. Kukimoto and R. K. Brayton, "Exact required time analysis via false path detection," in *Proc. IEEE Design Automation Conf.*, 1997, pp. 220–225.
[54] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*. New York: Wiley, 1990.
[55] R. Nair, C. L. Berman, P. S. Hauge, and E. J. Yoffa, "Generation of performance constraints for layout," *IEEE Trans. Computer-Aided Design*, vol. 8, no. 8, pp. 860–874, 1989.
[56] R. B. Hitchcock, Sr., G. L. Smith, and D. D. Cheng, "Timing analysis of computer hardware," *IBM J. Res. Develop.*, vol. 26, no. 1, pp. 100–108, 1992.
[57] M. Sarrafzadeh, D. Knol, and G. Tellez, "Unification of budgeting and placement," in *Proc. IEEE Design Automation Conf.*, 1997, pp. 758–761.
[58] R. H. J. M. Otten and R. K. Brayton, "Planning for performance," in *Proc. IEEE Design Automation Conf.*, 1998, pp. 122–127.
[59] D. Sylvester and K. Keutzer, "Getting to the bottom of deep submicron," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1998, pp. 203–211.
[60] D. Sylvester and K. Keutzer, "Getting to the bottom of deep submicron II: A global wiring paradigm," in *Proc. ACM Int. Symp. Physical Design*, 1999, pp. 193–200.
[61] R. S. Tsay and J. Koehl, "An analytical net weighting approach for performance optimization in circuit placement," in *Proc. ACM/IEEE Design Automation Conf.*, 1991, pp. 620–625.
[62] H. Youssef, R.-B. Lin, and S. Shragowitz, "Bounds on net delays," *IEEE Trans. Circuits Syst.*, vol. 39, no. 11, pp. 815–824, 1992.
[63] V. K. R. Chiluvuri and I. Koren, "Layout-synthesis techniques for yield enhancement," *IEEE Trans. Semiconduct. Manufact.*, vol. 8, pp. 178–187, 1995.
[64] W. B. Glendinning and J. N. Helbert, *Handbook of VLSI Microlithography: Principles, Technology, and Applications*: Noyes, 1991.
[65] A. B. Kahng, S. Muddu, E. Sarto, and R. Sharma, "Interconnect tuning strategies for high-performance ICs," in *Proc. Conf. Design Automation Test Eur.*, 1998, pp. 471–478.
[66] A. B. Kahng, S. Muddu, and D. Vidhani, "Noise and delay uncertainty studies for coupled RC interconnects," in *Proc. IEEE Int. ASIC/SOC Conf.*, 1999, pp. 3–8.
[67] A. B. Kahng and Y. C. Pati, "Subwavelength lithography and its potential impact on design and EDA," in *Proc. ACM/IEEE Design Automation Conf.*, 1999, pp. 799–804.
[68] A. B. Kahng, G. Robins, A. Singh, H. Wang, and A. Zelikovsky, "Filling and slotting: Analysis and algorithms," in *Proc. Int. Symp. Physical Design*, 1998, pp. 95–102.
[69] A. B. Kahng, H. Wang, and A. Zelikovsky, "Automated layout and phase assignment techniques for dark field alternating PSM," in *Proc. SPIE 18th Annu. BACUS Symp. Photomask Technology*, 1998, pp. 222–231.
[70] D. A. Kirkpatrick, "The implications of deep sub-micron technology on the design of high performance digital VLSI systems," Ph.D. dissertation, Univ. California, Berkeley, 1997.
[71] M. D. Levenson, "Wavefront engineering from 500 nm to 100 nm CD," *Proc. SPIE*, vol. 3049, pp. 2–13, 1997.
[72] M. D. Levenson, N. S. Viswanathan, and R. A. Simpson, "Improving resolution in photolithography with a phase-shifting mask," *IEEE Trans. Electron Devices*, vol. ED-29, pp. 1828–1836, 1982.
[73] L. W. Liebmann, T. H. Newman, R. A. Ferguson, R. M. Martino, A. F. Molless, M. O. Neisser, and J. T. Weed, "A comprehensive evaluation of major phase shift mask technologies for isolated gate structures in logic designs," *Proc. SPIE*, vol. 2197, pp. 612–623, 1994.
[74] H.-Y. Liu, L. Karklin, Y.-T. Wang, and Y. C. Pati, "The application of alternating phase-shifting masks to 140 nm gate patterning: Line width control improvements and design optimization," in *SPIE 17th Annu. BACUS Symp. Photomask Technology*, vol. 3236, 1998, pp. 328–337.
[75] W. Maly, "Computer-aided design for VLSI circuit manufacturability," *Proc. IEEE*, vol. 78, pp. 356–392, 1990.
[76] ——, "Moore's law and physical design of ICs," in *Proc. ACM Int. Symp. Physical Design*, 1998.
[77] J. Nistler, G. Hughes, A. Muray, and J. Wiley, "Issues associated with the commercialization of phase shift masks," in *SPIE 11th Annu. BACUS Symp. Photomask Technology*, vol. 1604, 1991, pp. 236–264.
[78] P. Rai-Choudhury, *Handbook of Microlithography, Micromachining, and Microfabrication—Vol. 1: Microlithography*: SPIE Opt. Eng., 1997.
[79] SEMATECH, "Workshop notes," in *3rd SEMATECH Litho-Design Workshop*, 1996.
[80] Semiconductor Industry Association, *The National Technology Roadmap for Semiconductors*, 1997.
[81] B. E. Stine, D. S. Boning, J. E. Chung, and L. Camilletti, "The physical and electrical effects of metal-fill patterning practices for oxide chemical-mechanical polishing processes," *IEEE Trans. Electron Devices*, vol. 45, pp. 665–679, 1998.
[82] B. E. Stine, V. Mehrotra, D. S. Boning, J. E. Chung, and D. J. Ciplickas, "A simulation methodology for assessing the impact of spatial/pattern dependent interconnect parameter variation on circuit performance," in *IEDM Tech. Dig.*, 1997, pp. 133–136.
[83] T. Waas, H. Hartmann, and W. Henke, "Automatic generation of phase shift mask layouts," *Microelectron. Eng.*, vol. 23, pp. 139–142, 1994.
[84] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Englewood Cliffs, NJ: Prentice-Hall, 1979.
[85] E. Nequist and L. Scheffer, "Why interconnect prediction doesn't work," in *Proc. ACM Int. Workshop System-Level Interconnect Prediction*, 2000, pp. 139–144.
[86] A. Krstic and K.-T. Cheng, *Delay Fault Testing for VLSI Circuits*. Norwell, MA: Kluwer, 1998.
[87] K. T. Cheng and A. Krstic, "Current directions in automatic test-pattern generation," *IEEE Computer*, pp. 58–64, 1999.
[88] J. L. Huang and K. T. Cheng, "A sigma-delta modulation based BIST scheme for mixed-signal circuits," in *Proc. Asia South Pacific Design Automation Conf.*, 2000, pp. 605–610.
[89] J. L. Huang, C. K. Ong, and K. T. Cheng, "A BIST scheme for on-chip ADC and DAC testing," in *Proc. Design Automation Test Eur.*, 2000, pp. 216–220.
[90] J.-J. Liou, K.-T. Cheng, and D. Mukherjee, "Path selection for delay testing of deep sub-micron devices using statistical performance sensitivity analysis," in *Proc. IEEE VLSI Test Symp.*, 2000, pp. 97–104.
[91] W.-C. Lai, A. Krstic, and K.-T. Cheng, "On testing the path delay faults of a microprocessor using its instruction set," in *Proc. IEEE VLSI Test Symp.*, 2000, pp. 15–20.

[92] W.-C. Lai, A. Krstic, and K.-T. Cheng, "Test program synthesis for path delay faults in microprocessor cores," in *Proc. Int. Test Conf.*, 2000, pp. 1080–1089.
[93] L. Chen, S. Dey, P. Sanchez, K. Sekar, and Y. Chen, "Embedded hardware and software self-testing methodologies for processor cores," in *Proc. ACM/IEEE Design Automation Conf.*, 2000, pp. 625–630.
[94] B. Dufort and G. W. Roberts, "Signal generation using periodic single and multi-bit sigma-delta modulated streams," in *Proc. Int. Test Conf.*, 1997, pp. 396–405.
[95] G. Moore, "VLSI: Some fundamental changes," *IEEE Spectr.*, vol. 16, no. 4, pp. 79–37, 1979.
[96] W. Maly, "High levels of IC manufacturability: One of the necessary prerequisites of the 1997 SIA roadmap," in *Proc. IEDM*, 1998, pp. 759–762.
[97] ——, "Computer-aided design for VLSI circuit manufacturability," *Proc. IEEE*, vol. 78, no. 2, pp. 356–392, 1990.
[98] W. Maly, H. Heineken, J. Khare, and P. K. Nag, "Design for manufacturability in submicron domain," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1996, pp. 690–697.
[99] ——, "Design-manufacturing interface: Part I—Vision," in *Proc. Design, Automation Test Eur.*, 1998, pp. 550–556.
[100] W. Maly, H. T. Heineken, J. Khare, P. K. Nag, P. Simon, and C. Ouyang, "Design-manufacturing interface: Part II—Applications," in *Proc. Design, Automation Test Eur.*, 1998, pp. 557–562.
[101] H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*. Reading, MA: Addison-Wesley, 1990.
[102] W. Maly, H. Jacobs, and A. Kersch, "Estimation of wafer cost for technology design," in *Proc. IEDM*, 1993, pp. 35.6.1–35.6.4.
[103] W. Maly, "Cost of silicon viewed from VLSI design perspective," in *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 135–142.
[104] P. K. Nag, W. Maly, and H. Jacobs, "Simulation of yield/cost learning curves with Y4," *IEEE Trans. Semiconduct. Manufact.*, pp. 256–266, May 1997.
[105] H. T. Heineken and W. Maly, "Performance-manufacturability trade-offs in IC design," in *Proc. Design, Automation Test Eur.*, 1998, pp. 563–567.
[106] J. Khare, W. Maly, S. Griep, and D. Schmitt-Landsiedel, "Yield-oriented computer-aided defect diagnosis," *IEEE Trans. Semiconduct. Manufact.*, vol. 8, no. 2, pp. 195–205, 1995.
[107] W. Maly, "Keynote: Future of testing: Reintegration of design, testing and manufacturing," in *Proc. Eur. Design Test Conf.*, 1996, pp. xix–xxii.
[108] ——, "Keynote: Testing-based failure analysis: A critical component of the SIA roadmap vision," in *Proc. 23rd Int. Symp. Testing Failure Analysis*, Oct. 1997, pp. 3–6.

Randal E. Bryant (Fellow, IEEE) received the B.S. degree in applied mathematics from the University of Michigan, Ann Arbor, in 1973, and the S.M., E.E., and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1977, 1978, and 1981, respectively.

He was on the Faculty at the California Institute of Technology from 1981 to 1984. Since September 1984, he has been at Carnegie Mellon University, Pittsburgh, PA, and is currently the President's Chair of Computer Science and has been Head of the Computer Science Department.

Dr. Bryant received the 1987 IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN Best Paper Award, the 1989 IEEE Baker Prize, and the ACM Kanellakis Theory and Practice Award. He was an Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN from 1989 to 1995 and Editor-in-Chief from 1995 to 1997. He is a Fellow of the ACM.

Kwang-Ting (Tim) Cheng (Fellow, IEEE) received the B.S. degree in electrical engineering from the National Taiwan University in 1983, and the Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley, in 1988.

From 1988 to 1993, he was with AT&T Bell Laboratories, Murray Hill, NJ. In 1993, he joined the faculty at the University of California, Santa Barbara, where he is currently Professor of Electrical and Computer Engineering and Director of Computer Engineering Program. He has authored or coauthored more than 150 technical papers, coauthored three books, and holds nine U.S. patents. His current research interests include VLSI testing, design synthesis, and design verification.

Dr. Cheng received Best Paper awards at the 1994 and the 1999 Design Automation Conference and the 1987 AT&T Conference on Electronic Testing. He is currently an Associate Editor-in-Chief for IEEE DESIGN and TEST of COMPUTERS. He also serves on the Editorial Boards of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN and the JOURNAL OF ELECTRONIC TESTING: THEORY AND APPLICATIONS. He served as General Chair and Program Chair of IEEE International Test Synthesis Workshop and has also served on the technical program committees for several international conferences on CAD and testing, including DAC, ICCAD, ITC, and VTS.

Andrew B. Kahng received the A.B. degree in applied mathematics (physics) from Harvard University, Cambridge, MA, and the M.S. and Ph.D. degrees in computer science from the University of California, San Diego.

From 1983 to 1986, he was with Burroughs Corporation Micro Components Group, San Diego, where he worked in device physics, circuit simulation, and CAD for VLSI layout. He joined the Computer Science Department, University of California, Los Angeles (UCLA), as an Assistant Professor in 1989 and became Associate Professor in 1994 and Professor in 1998. From 1996 through 1997, he was a Visiting Scientist at Cadence Design Systems, Inc. He resumed his duties at UCLA in 1997 and from 1998 to 2000, served as the Computer Science Department's Vice-Chair for Graduate Studies. He is currently with the University of California, San Diego, as a Professor in the CSE and ECE Departments. His research interests include VLSI physical layout design and performance analysis, combinatorial and graph algorithms, stochastic global optimization, and the optimization foundations of computational commerce.

Prof. Kahng received the National Science Foundation Research Initiation and Young Investigator Awards, six Best Paper nominations, and a DAC Best Paper award. He was the founding General Chair of the 1997 ACM/IEEE International Symposium on Physical Design and Cofounder of the ACM Workshop on System-Level Interconnect Prediction. He defined the physical design roadmap as a Member of the Design Tools and Test Technical Working Group for the 1997, 1998, and 1999 renewals of the SIA Technology Roadmap for Semiconductors. He has also served as a Member of the EDA Council's 200× Task Force. Currently, he is Chair of the U.S. Design TWG for the 2001 renewal of the International Technology Roadmap for Semiconductors and Technical Program Chair of the 2001 Electronic Design Processes Symposium of the IEEE DATC. He is also on the steering committees of ISPD-2001 and SLIP-2001.

Kurt Keutzer received the B.S. degree in mathematics from Maharishi International University, Fairfield, IA, in 1978, and the M.S. and Ph.D. degrees in computer science from Indiana University, Bloomington, in 1981 and 1984, respectively.

In 1984, he joined AT&T Bell Laboratories, where he worked to apply various computer science disciplines to practical problems in computer-aided design. In 1991, he joined Synopsys, Inc., where he continued his research that culminated in his position as Chief Technical Officer and Senior Vice-President of Research. Since 1988, he has been a Professor of Electrical Engineering and Computer Science at the University of California, Berkeley. He coauthored *Logic Synthesis* (New York: McGraw-Hill, 1994). His current research interests include synthesis and high-level design.

Prof. Keutzer has received three Design Automation Conference (DAC) Best Paper Awards, a Distinguished Paper Citation from the International Conference on Computer-Aided Design (ICCAD), and a Best Paper Award at the International Conference in Computer Design (ICCD). From 1989 to 1995, he was an Associate Editor of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN and is currently on the editorial boards of three journals: *Integration—the VLSI Journal*, *Design Automation of Embedded Systems*, and *Formal Methods in System Design*. He has served on the technical program committees of DAC, ICCAD, and ICCD as well as the technical and executive committees of numerous other conferences and workshops.

Wojciech Maly (Fellow, IEEE) received the M.Sc. degree in electronic engineering from the Technical University of Warsaw, Poland, in 1970, and the Ph.D. degree from the Institute of Applied Cybernetics, Polish Academy of Sciences, Warsaw, Poland, in 1975.

From 1970 to 1973, he was with the Institute of Applied Cybernetics. In 1973, he was with the Technical University of Warsaw and became an Assistant Professor in 1975. From 1979 to 1981, he was a Visiting Assistant Professor of Electrical and Computer Engineering at Carnegie Mellon University, Pittsburgh, PA. Since September 1983, he has been with Carnegie Mellon University, where he is currently a Whitaker Professor of Electrical and Computer Engineering. He has authored, coauthored, or edited a number of books, journals, and conferences papers as well as patents, which have attempted to promote integration of design, test, and manufacturing. His research interests have focused on the interfaces between VLSI design, testing, and manufacturing with the stress on the stochastic nature of phenomena relating these three VLSI domains.

Dr. Maly has received various awards, including honors for his Ph.D. thesis, Ministry of Higher Education of Poland Research Award, Carnegie Mellon's Benjamin Richard Teare Teaching Award, AT&T Foundation Award for Excellence in Instructing of Engineering Students, Fellowship from Deutsche Forschungsgemeinschaft, SRC 1992 Technical Excellence Award, the 1990 International Test Conference Best Paper Award, the 1994 ESREF Best Paper Award, the 1994 IEEE TRANSACTIONS ON SEMICONDUCTOR MANUFACTURINGBest Paper Award, 1995 Best Paper Award from the European Design and Test Conference, and the Eta Kappa Nu CMU Sigma Chapter Excellence in Teaching Award.

Richard Newton (Fellow, IEEE) received the B.Eng. and M.Eng.Sci. degrees from the University of Melbourne, Australia, in 1973 and 1975, respectively, and the Ph.D. degree from the University of California, Berkeley, in 1978.

He is currently a Professor and Chair of the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. Since 1979, he has been actively involved as a Researcher and Teacher in the areas of design technology, electronic system architecture, and integrated circuit design. From 1986 to 1988, he was Vice Chairman for Computing Resources in the Department of Electrical Engineering and Computer Sciences. In addition to his academic role, he has helped to found a number of design technology companies, including SDA Systems (now Cadence Design Systems), PIE Design Systems (now Quickturn), Simplex Solutions, Crossbow, and Synopsys, where he recently rejoined the Board of Directors. He was also a Founder and Director of nChip (now a part of Flextronics, Inc.), Aptix, and Objectivity, and was a Director of Interconnectix (now a Mentor Graphics company). Since 1988, he has been a Venture Partner with the Mayfield Fund, a high-technology venture capital partnership in which he has contributed to both the evaluation and early-stage development of over a dozen new companies, for three of which he is currently a Member of the Board of Directors. From 1994 to 1995, he was the acting President and CEO of Silicon Light Machines (formerly Echelle, Inc.), a development-stage company that brings to market a number of display systems based on the application of micromachined silicon light-valves. Since 1997, he has been a member of the Technical Advisory Board of the Microsoft Research Laboratories. He was a Founding Member of both the EDIF technical and steering committees, an Advisor to the CAD Framework Initiative, and a Founding Member of EDAC.

Prof. Newton has received a number of awards for his teaching and research, including Best Paper Awards at the 1988 European Solid-State Circuits Conference, the 1987 and 1989 ACM/IEEE Design Automation Conferences, and the International Conference on Computer Design and the 1989 IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN Best Paper Award. He received the 1987 C. Holmes McDonald Outstanding Young Professor Award of the Eta Kappa Nu Engineering Honor Society. He was an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN from 1984 to 1988 and a Member of the Administrative Committee of the IEEE Circuits and Systems Society from 1985 to 1988. He is a Member of the ACM.

Lawrence Pileggi (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 1989.

He is currently a Professor of Electrical and Computer engineering at Carnegie Mellon University. From 1984 through 1986, he was with Westinghouse Research and Development, where he was recognized with the corporation's highest engineering achievement award in 1986. From 1989 to 1995, he was a faculty member at the University of Texas, Austin. He has consulted for several EDA and semiconductor companies and, while on leave from Carnegie Mellon from 1999 to 2000, he was the Chief Technical Officer and Vice President of Research and Design at Monterey Design Systems. He is a coauthor of *Electronic Circuit and System Simulation Methods* (New York: McGraw-Hill, 1995). He has authored or coauthored over 100 refereed conference and journal papers, and holds five U.S. patents. His current research interests include various aspects of circuit-level design automation and analysis.

Prof. Pileggi received the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN Best Paper Award in 1991 for "Asymptotic Waveform Evaluation (AWE)" and in 1999 for "Passive Reduced-Order Interconnect Macromodeling Algorithm (PRIMA)." He received a Presidential Young Investigator Award from the National Science Foundation in 1991. In 1991 and 1999, he received the Semiconductor Research Corporation Technical Excellence Award. In 1993, he received an Invention Award from the SRC and, subsequently, a U.S. Patent for the RICE simulation tool. In 1994, he received The University of Texas Parent's Association Centennial Teaching Fellowship for Excellence in Undergraduate Instruction. In 1995, he received a Faculty Partnership Award from IBM.

Jan M. Rabaey (Fellow, IEEE) received the E.E. and Ph.D. degrees in applied sciences from the Katholieke Universiteit Leuven, Belgium, in 1978 and 1983, respectively.

From 1983 to 1985, he was with the University of California, Berkeley, as a Visiting Research Engineer. From 1985 to 1987, he was a Research Manager at IMEC, Belgium. He has been with the Electrical Engineering and Computer Science Department of the University of California, Berkeley since 1987, where he is currently a Professor and Associate Chair of the department and the Scientific Co-Director of the Berkeley Wireless Research Center (BWRC). He also has been a Visiting Professor at the University of Pavia, Italy, and Waseda University, Tokyo, Japan. He has authored or coauthored a wide range of papers and books in the area of signal processing, digital architectures, and design automation. His current research interests include the conception and implementation of next-generation integrated wireless systems, including the analysis and optimization of communication algorithms and networking protocols, the study of low-energy implementation architectures and circuits, and the supporting design automation environments.

Prof. Rabaey has received numerous scientific awards, including the 1985 IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN Best Paper Award, the 1989 Presidential Young Investigator Award, and the 1994 Signal Processing Society Senior Award. He was Chair of the VLSI Signal Processing Technical Committee of the Signal Processing Society, the International Symposium on Low Power Electronics, and the IFIP Conference on Mobile Computing in 1996. He is currently the General Chair of the Design Automation Conference.

Alberto Sangiovanni-Vincentelli (Fellow, IEEE) received the Dottore in Ingegneria degree in electrical engineering and computer science from the Politecnico di Milano, Milano, Italy in 1971.

He is currently the Edgar L. and Harold H. Buttner Chair of Electrical Engineering and Computer Sciences, University of California, Berkeley, where he has been on the Faculty since 1976. In 1980 to 1981, he was as a Visiting Scientist at the Mathematical Sciences Department, IBM T. J. Watson Research Center. In 1987, he was Visiting Professor at the Massachusetts Institute of Technology, Cambridge, and has held a number of Visiting Professor positions at the University of Torino, University of Bologna, University of Pavia, University of Pisa, and University of Rome. He is a Cofounder of Synopsys, where he was also Chair of the Technical Advisory Board, and Cadence—the two leading companies in the area of electronic design automation. He was also a Director of ViewLogic and Pie Design Systems, and is currently the Chief Technology Advisor of Cadence Design Systems. He is the Founder of the Cadence Berkeley Laboratories, the Cadence European Laboratories, and the Kawasaki Berkeley Concept Research Center, where he is currently Chairman of the Board. He is the Scientific Director of the Project on Advanced Research on Architectures and Design of Electronic Systems (PARADES), a European Group of Economic Interest supported by Cadence, Magneti-Marelli, and ST Microelectronics. He is a Member of the Board of Directors of Cadence, Sonics Inc., and Accent, an ST Microelectronics-Cadence joint venture, is on the Advisory Board of the Lester Center of the Haas School of Business and of the Center for Western European Studies, and is a Member of the Berkeley Roundtable of the International Economy (BRIE). In addition, he has consulted for a number of U.S. companies, Japanese companies, and European companies. He has authored or coauthored over 520 papers and ten books in the area of design methodologies, large-scale systems, embedded controllers, hybrid systems, and tools.

Dr. Sangiovanni-Vincentelli received the Distinguished Teaching Award of the University of California, the 1995 Graduate Teaching Award of the IEEE, the 1982–1983 Guillemin-Cauer Award, the 1987–1988 Darlington Award, and the 1999 CASS Golden Jubilee Medal. He was the Technical Program Chairperson of the International Conference on Computer-Aided Design and the Executive Vice-President of the IEEE Circuits and Systems Society. He is a Member of the National Academy of Engineering.