The Designer's Guide Community downloaded from www.designers-guide.org

Design of Mixed-Signal Systems on Chip

Ken Kundert, Henry Chang, Dan Jefferies, Gilles Lamant, Enrico Malavasi, Fred Sendig

Version 1a, December 2000

The electronics industry is increasingly focused on the consumer marketplace, which requires low-cost high-volume products to be developed very rapidly. This, combined with advances in deep sub-micron technology have resulted in the ability and the need to put entire systems on a single chip. As more of the system is included on a single chip, it is increasingly likely that the chip will contain both analog and digital sections. Developing these mixed-signal systems-on-chip presents enormous challenges both to the designers of the chips and to the developers of the CAD systems that are used during the design process. This paper presents many of the issues that act to complicate the development of large single-chip mixed-signal systems and how CAD systems are expected to develop to overcome these issues.

Published in the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, *vol. 19, no. 12, pp. 1561-1571, December 2000.*

Last updated on May 11, 2006. You can find the most recent version at www.designers-guide.org. Contact the author via e-mail at ken@designers-guide.com.

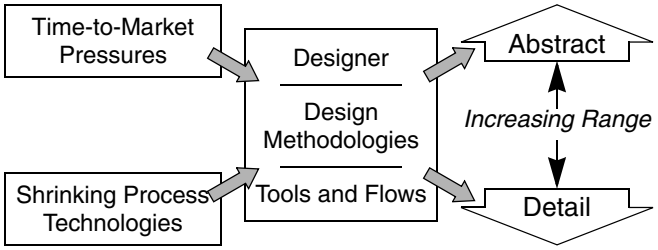
Permission to make copies, either paper or electronic, of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage and that the copies are complete and unmodified. To distribute otherwise, to publish, to post on servers, or to distribute to lists, requires prior written permission.

Copyright © 2006, Kenneth S. Kundert – All Rights Reserved

1.0 Introduction

Increasing time-to-market (TTM) pressures due to the continued consumerization of the electronics market place and the availability of shrinking process technologies are the two fundamental forces driving designers, design methodologies, and EDA tools and flows today. This is illustrated in Figure 1

FIGURE 1. *Design drivers and design methodology gaps.*



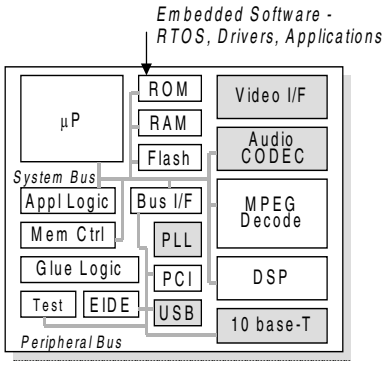
On one hand, TTM pressures, along with the added integration afforded by newer process technologies, have forced a move to higher levels of abstraction to cope with the added complexity in design. This can already be seen in the digital design domain space, where cell based design is rapidly moving to Intellectual Property (IP), re-use based or block-based design methodologies [4]. On the other hand, shrinking process technologies have also caused a move in the opposite direction: because of the increasing significance of physical effects, there has been a need to observe lower levels of detail. Signal integrity, electro-migration, and power analysis are now adding severe complications to design methodologies already stressed by the increasing device count. This is true for both analog and digital design. The total range of design abstractions encountered in a single design flow is continually growing, and pulling in opposite directions (abstraction vs. detail). Managing this increasing range, and insuring that the system definitions (constraints) are preserved and verified (or verifiable) through all levels of abstractions and between different levels is where one becomes acutely aware of the widening gaps in today's design methodologies. However, to meet TTM needs, it is imperative that these be kept under control.

The stresses caused by this wide abstraction range and the increasing complexity of design at each level of abstraction uncover significant methodology gaps. These occur both between abstraction layers as well as within them. Design methodologies, tools, and flows, evolve to try to hold the design "system" together. However, what we see today is just the beginning of what is to come, with the new, even smaller, process technologies.

Stressed by cost and performance objectives resulting from the consumerization of electronics, designers are driven to take advantage of the smaller process technologies, putting entire systems on chips. Two basic types of systems-on-a-chip (SOC) exist — one that has grown from the ASIC world, and the other from the custom IC world. An example of the former is shown in Figure 2 This is a design that is mostly digital. It is a programmable system that integrates most of the functions of the end product. It contains processors. It has embedded software, peripherals both analog and digital, and has a bus-based architecture. Analog and mixed-signal design blocks are only integrated if

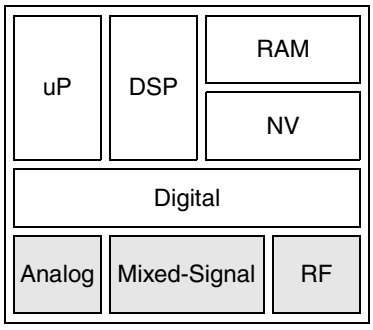
they can be in a reasonable time and at a reasonable cost. For example, high-frequency RF remains as a separate chip for this type of design. For this type of design, the integrator is a digital designer and increasingly, the cost is in the development of the embedded software rather than in the hardware design of the IC.

FIGURE 2. *An ASIC-SOC example.*



The other type of design, which we will henceforth refer to as *AMS-SOC*, is shown in Figure 3 This is a design that began in the realm of custom mixed-signal designs. These are designs that are both high in performance and have complex signal paths through both analog and digital components. Examples of these designs include PRML disk drive controllers, xDSL front-ends, 10/100 base-T physical layers and RF front-ends. This era of process technology has also allowed analog and mixed-signal designers to begin to integrate significant amounts of the functionality of the entire systems onto a single chip. However, unlike the case of the ASIC design moving to SOC, the analog/mixed-signal design is not an "option." It is the critical and probably the differentiating part of the ICs with the digital part optional as to whether or not it is integrated. In this case, at today's process technology, embedded software is not yet a significant issue. The most significant issues lie around the design and integration of digital and analog/mixed-signal blocks.

FIGURE 3. *An AMS-SOC example.*

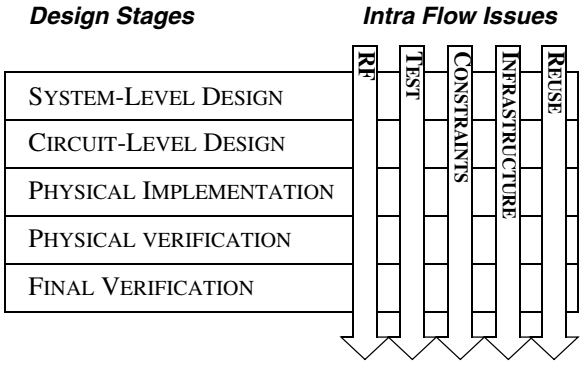


These are the designs that are the main focus of this paper. The highest level of abstraction is the system level. Thus, in this case, the range of abstraction levels spans from the device level (including parasitic devices) through to the system level.

Due to the complex feedback loops that involve signal paths crossing the interface between digital and analog blocks multiple times, as well as less obvious physical effects between the analog and digital blocks, we believe that we are reaching a point where ad-hoc "patching" of the design process will not hold it together anymore and allow meeting TTM objectives for this type of design.

The design methodology needed for the design of AMS-SOCs dictates a design flow that can be broken down into a set of design stages as shown in FIGURE 4. Section 2.0 explores this design methodology and each of the design stages. But not all aspects of

FIGURE 4. *The design flow.*



design can be neatly separated into these stages. There are certain design capabilities and tool requirements that span the design process. In many ways these are the more difficult for EDA tool vendors to address as they are not contained wholly within the expertise of a specific design stage, and of necessity require interaction across the designers and design tools at each of these stages. Section 3.0 explores these complex intra flow issues and how they might be addressed.

2.0 The Design Flow

In this section we analyze the main areas that must be addressed to provide a workable solution to the problem of developing successful AMS-SOC designs.

The solution must consist of a set of design methodologies, tool flows, as well as an appropriate and cohesive set of tools. All of these are necessary to create a complete solution. While a specific design group may not have the desire or need for all three, all three must be considered in concert to develop the complete solution.

To provide a framework for what the solution entails, we have selected specific aspects of the design process and provided an overview of some of what is needed in each area.

The design flow of a complex AMS-SOC starts with an idea and ends with a layout. In between is a series of refinement and verification steps. First the idea is refined to a

series of specifications, which are verified by talking to potential customers. Then the specifications are refined to a functional description or an algorithm, which is verified with system simulators. The functional description is refined to an architecture, which is verified by simulators that interpret mixed-signal hardware description languages (MS-HDL, see Section 3.0-3.5). The blocks are then refined to the transistor level and are verified with timing simulators or with SPICE. This represents the electrical design process.

A similar process occurs for the physical design. The architecture is converted to a floorplan, which is then refined until the blocks are laid-out and routed. Verification of the layout involves checking the layout to assure it matches the schematic and that it satisfies all manufacturability rules. Final verification involves extracting the circuit from the layout, including layout parasitics, and simulating it with transistor-level simulators.

In high performance analog and MS designs the physical implementation often has such an impact on the circuit performance that circuit and layout issues must be considered together [44]. As a consequence the physical design is intertwined with circuit design and optimization, and the physical implementation is subject to frequent extraction, analyses and engineering change orders (ECO's), or incremental modifications. These frequent disruptions of the design flow are characteristic of AMS circuit development, and often account for a good portion of the overall time to market.

Once layout is complete, the whole design is represented in fine detail and the simulations are quite expensive. This prevents all but basic functionality from being verified at this point. Thus, the design process itself must assure, with a high degree of confidence, that the design functions properly in all situations and meets its performance requirements. This requires that:

- A formal verification plan be developed and followed throughout the design process [18]. The plan must assure that the design be verified continually along the design process.
- The ability to co-simulate blocks at different levels of abstraction so that the design can be continuously verified as it progresses from an abstract to detailed levels of representation.
- Constraint definition, translation, and verification from the architecture level through functional, circuit and physical levels.
- Reliable and easy communication of connectivity, constraints, parasitics and models between systems, circuit and layout designers.
- Extraction of models for each block that faithfully represent its behavior and performance as implemented. These models are used with system or HDL simulators to verify the design from the bottom up.

2.1 Top-Down Design

Most analog chips at one time were designed to be general-purpose building blocks optimized for performance, cost, or low power dissipation. This involved precision work at the transistor level by a specialist. Design exploration and circuit function and performance verification occurred more or less together. For small performance-critical analog and mixed-signal ICs, this remains the dominant design style. For large designs, however, this approach has several problems, including at least two fundamental ones:

1. Simulations take so long that comprehensive analysis of the design in manageable time frames becomes problematic. Because of this, projects may be delayed because of the need for extra silicon prototypes caused by inadequate verification.
2. For large designs, improvements made at the architectural level generally provide the greatest impact on the performance, cost and functionality of the chip. By the time the development reaches the circuit level, meaningful improvements are often very expensive.

In order to address these challenges, many design teams are either looking to, or else have already implemented, top-down methodologies [3]. In a basic top-down approach, the architecture of the chip is defined as a block diagram and simulated and optimized using either a MS-HDL simulator or a system simulator. From the high-level simulation, requirements for the individual circuit blocks are derived. Circuits are then designed individually to meet these specifications. Finally, the entire chip is laid out and verified against the original requirements.

A few of the key characteristics of these design styles are:

- Design exploration and verification are somewhat separate. The combination of greater simulation speed from the use of high-level behavioral models and the ability to perform parametric design make MS-HDL simulation appropriate for design exploration. The use of transistor-level simulation becomes more focused on verifying that the blocks match the intent of the high-level design.
- Parametric design at the system level. MS-HDLs provide users great flexibility in modeling. However, since a fundamental objective of the block-level analysis is to develop specifications for the block implementation, good top-down practice is to write the MS-HDL models so that their key performance characteristics are specified using parameters and so can be easily adjusted.
- Mixed-level simulation. In general, it is much faster to verify the functionality and performance of a specific block against its specifications within an MS-HDL representation of the system than it is to verify the entire design “flat” at the circuit level.

In practice, a final verification of the entire design at the circuit level, in SPICE, may still be desirable for verification of connectivity, proper startup, and the performance of critical paths. However, a major objective of most top-down approaches is to eliminate the need to do this more than once per project.

These practices require substantial attention early in the design process. This is the essential trade-off of top-down methodologies: more analysis early in design to avoid problems later on.

The main objectives of top-down approaches are to optimize globally the performance of the design, and to increase the general predictability of the design schedule. They also make it easier to coordinate the efforts of multiple designers working in parallel on different parts of the design at once.

The principal drawbacks are the need for rigor in the design process, and the need for designers to learn an MS-HDL, which presently few have significant familiarity with. Some of the early proprietary languages acquired a perhaps-justified reputation as difficult to learn and use. However, modern MS-HDL's like Verilog-AMS are better. Furthermore, our experience is that the effort required to make MS-HDL models is not only worthwhile, but also drops dramatically over the first few projects, as engineers learn the methodology and begin to reuse their existing models.

Top-down design represents a substantial shift from the way most people design today and there is considerable inertia that acts to slow its adoption. Those that have moved to a top-down design style have seen dramatic improvements in time-to-market and the ability to handle complexity. The best way to overcome the inertia that prevents top-down design from being adopted is to teach the art of top-down design and behavioral modeling in the universities.

2.2 System-Level Design

System-level design is generally performed by system architects. Their goal is to find an algorithm and architecture that implement the required functionality while providing adequate performance at minimum cost. They use system-level simulators, such as Matlab [26] or SPW [48], that allow them to explore various algorithms and evaluate tradeoffs quickly. These tools are preferred because they represent the design as a block diagram and have large libraries of predefined blocks for common application areas.

Once the algorithm is chosen, it must be mapped to a particular architecture. Thus, it must be refined to the point where the blocks used at the system level accurately reflect the way the circuit is partitioned for implementation. The blocks must represent sections of the circuit that are to be designed and verified as a unit. Furthermore, the interfaces must be chosen carefully to avoid interaction between the blocks that are hard to predict and model, such as loading or coupling. The primary goal at this phase is the accurate modeling of the blocks and their interfaces. This contrasts with the goal during algorithm design, which is to quickly predict the output behavior of the entire circuit with little concern about matching the architectural structure of the chip as implemented. As such, mixed-signal hardware description languages (MS-HDLs) such as Verilog-AMS [51] or VHDL-AMS [6,22,52] become preferred during this phase of the design because they allow accurate modeling of the interfaces and support mixed-level simulation (discussed in Section 2.4).

The transition between algorithm and architecture design currently represents a discontinuity in the design flow. The tools used during algorithm design are different from the ones used during architecture design, and they generally operate off of different design representations. Thus, the design must be re-entered, which is a source of inefficiencies and errors. It also prevents the test benches and constraints used during the algorithm design phase from being used during the rest of the design.

On the digital side, tools such as SPW do provide paths to implementation via Verilog and VHDL generation. However, as of today, they have yet to be tightly integrated into the remainder of the design flow. Similar capabilities do not yet exist for the analog or mixed-signal portions of the design. An alternative is to use Verilog-AMS or VHDL-AMS for both algorithm and architecture design. This has not been done to date because simulators that support these languages are just now becoming available. As such, there is a dearth of application specific libraries.

2.3 Analog Synthesis

The ability to automatically convert a high-level specification of a block to a circuit-level implementation is referred to as synthesis. While synthesis is well established in the digital world, for analog or mixed-signal circuits it is only available in special cases, such as for filters. Research into analog synthesis has developed over the last two

decades in many directions, from early work on knowledge-based module compilation [2,8] to more recent optimization intensive approaches [7,29,36]. Optimization is based either on numerical simulation [35] or on analytic models [28]. To help in the development of analytic models, a significant research effort went into exploration of symbolic analysis [12,45]. Beyond model building, symbolic analysis was also applied to more ambitious goals, such as topology exploration with interesting results [27], whose applicability unfortunately is limited to selected categories of analog circuits.

Many attempts at building analog design automation systems have been made. The most important ones are probably ADAM [8,9], a commercial product developed at CSEM, and ACACIA [5,36], developed at CMU, which recently expanded to include, among other things, RF design [1]. Several good survey papers provide insight into the extensive research production in this field [13,43,44].

Some commercial offerings in this space have appeared recently. Noticeable among others are NeoCell™ [37], a system for analog cell design automation, which leverages in part from the technology developed for ACACIA; and Picasso Op-Amp™ [38] and Dali RF Tool Suite™ [39], web-based tools providing on-demand circuit topology selection and sizing based on geometric programming [17]. However, we believe that the large variety and complexity of analog cells makes it unlikely that a general solution for the problem of analog synthesis will be available in the near future. Instead, it is likely that a variety of design aids and very specific module generators will become available for an increasing variety of analog cells and blocks to ease the transition from high-level specification to circuit-level implementation.

2.4 Mixed-Level Simulation

Using a top-down design methodology is expected to become the norm for designing complex mixed-signal circuits. As such, the system architecture will be fully explored and verified using either a MS-HDL or a system simulator before individual blocks are designed. However, once the blocks are designed, they must be verified in the context of the system to assure that they will operate properly within the system. At this point, it must be possible to co-simulate behavioral models and transistor-level circuits together. The block-diagram used in the simulation of the architecture must be refined to the point where each block represents a relatively independent circuit that would be designed as a single unit. Pin-accurate MS-HDL models are developed for each block and the system is verified using these models.

The block designers then take the HDL models and a series of specifications as input and produce the transistor-level schematic and layout, which are passed back to the system engineers for integration and verification with the rest of the system. Using the ability to co-simulate transistor-level and behavioral-level descriptions of the blocks, the system is repeatedly verified by replacing one-by-one the HDL model of one block with the transistor-level implementation to verify the functionality of the block and its interfaces. This approach greatly reduces the cost of each simulation and increases the chance that miscommunications concerning block interfaces are caught early in the design process.

2.5 Physical Implementation

Physical implementation corresponds to a variety of tasks that can be grouped into two major areas:

- Block authoring
- Block/chip assembly

These two areas are deeply intertwined as most design flows require a mix of top-down and bottom-up approaches with a combination of soft and hard blocks, and behavioral, logical and physical representations of different parts [4](p.189ff.). Therefore any solution needs to incorporate a seamless flow including access to both authoring and assembly of complex blocks.

For block authoring successful commercial tools, methodologies and flows have been developed over the last few years. However the coordination of different design approaches into consistent flows and adequate solutions for block assembly is still under investigation, especially for mixed-signal applications in a SOC environment. Assembly and authoring need to be addressed simultaneously, since a design environment for large mixed-signal applications requires cooperation between an interactive editing environment and reentrant automation. A full custom implementation is also required for most analog portions of the design. Finally critical issues such as IP reuse [4][24], power management and signal integrity [49] are key to the success of large SOC designs.

A key component required to guarantee a good integration between assembly and authoring is the floorplanner. Commercial floorplanning technology today is focused on digital designs, and it is often poorly equipped to handle AMS issues such as noise and signal integrity. While encapsulation of AMS blocks is available in most commercial tools, it does not allow a correct representation of the interactions between blocks such as intermodulation, cross talk and substrate noise, and top-level mixed-signal interconnections require special modeling and planning [16]. The organization of power distribution is significantly different when analog and digital supply lines are used, with severe impact on substrate noise [46]. Finally, the design flow often follows a combination of top-down and bottom up steps very tightly interleaved.

For example, a design may be partitioned into various digital, analog, and mixed-signal blocks. Not only may these blocks be designed concurrently, they will be physically realized at different times and using different implementation flows. While large digital blocks can be created using a semi-automated design flow, other custom portions often need to be carefully hand crafted. Manual design does not scale well with circuit size, and custom blocks become the bottleneck of the entire flow, even though advanced assisted custom design methodologies have become available recently [33]. Finally, during chip assembly the communication between blocks will be subject to timing, power or signal integrity constraints. Coupling between blocks sometimes determines the inner structure of the blocks themselves. In order to meet these constraints, top-down modifications are forced upon the blocks. These operations must be kept consistent with the specific design flow, often bottom up, used for the authoring of each block. The need for reentrant and interoperable environments for the authoring of ASIC-style digital blocks, custom analog blocks, and the assembly of all these parts is a major paradigm shift that characterizes complex AMS-SOC's.

For the complex ASIC MS SOC designs, a physically aware automated synthesis to silicon flow such as being delivered as part of Envisia® PKS [15] may also be utilized.

This flow, while currently targeting complex deep sub-micron designs, needs to be extended in order to be able to read AMS designs. While not necessarily implementing the analog portions, this tool suite needs to become aware of them and to take them into account during physically aware synthesis.

2.6 Physical Verification

Very powerful technology for physical verification has been developed in recent years by the EDA industry. Commercial tools, such as Cadence's Assura®, Avant!'s Hercules® and Mentor Graphics's Calibre®, often include hierarchical capability for verification and extraction, and present various levels of integration with the block authoring tool suites. The resulting design cycle improvements have proven not only the importance of an efficient verification tool, but also the criticality of a solution flow where such a tool is well integrated with all the other applications:

- using a common database;
- using a common set of interactive commands for browsing and fixing errors;
- using a common user interface look-and-feel;
- supporting the same set of constraints.

Some of the proprietary netlist-based integration methodologies are shared by most commercial tools. Digital description languages used for simulation, such as Verilog and VHDL have been extended for AMS designs [22,51], and commercial verification tools will soon be required to support these AMS extensions.

From a strategic point of view, the verification phase must be tied more closely with the physical design cycle. New constraint-driven layout applications, able to enforce physical and electrical constraints, have recently become available [14]. These must be matched by corresponding new capabilities in the verification phase, which will have to become cognizant of the same set of AMS constraints. Substrate coupling noise verification, currently heavily limited for capacity reasons to circuit level within small blocks, must be used to optimize the distribution of guard rings and to drive block placement in mixed signal systems.

Yield has a parametric dependency on AMS performance functions and measurements, which can be captured through behavioral and stochastic models. The support of electrical and design constraints derived from these models will enable the physical verification phase to help in the design centering analysis.

With respect to manufacturing, the verification tools must also support the increasingly common post-layout processing techniques such as optical proximity correction (OPC) and new subwavelength lithography processes such as phase shift mask (PSM) [23].

2.7 Final Verification

Final verification is performed by using a physical verification tool to extract a netlist of the circuit, including parasitics, from the final layout. Of course, such circuits are very large and only limited verification is possible. With an AMS-SOC, it is often possible to do some transistor-level full chip simulation. Typically only areas of special concern that cannot be sufficiently verified using mixed-level simulation are considered. Examples include power-up behavior and timing of the critical paths.

In digital blocks, final verification is often performed using timing simulators, such as Synopsys's TimeMill or Avant!'s StarSim. Relative to circuit simulators such as Spice, timing simulators trade accuracy and generality for speed. They generally provide at least 10× in speed and capacity over SPICE but are suitable only for estimating the timing of MOS digital circuits, and can generally be counted on to produce timing numbers that are accurate to within 5% on these circuits. Analog circuits or circuits constructed with bipolar transistors often confuse timing simulators, causing them to run slowly and give incorrect results. Cadence's ATS overcomes this problem by combining a circuit simulator with a timing simulator and so can handle large digital MOS circuits that contain some analog or bipolar circuitry.

ASIC-SOCs are usually too large to be verified using any type of transistor-level simulation. Instead bottom-up verification is required. With bottom-up verification, individual blocks are extracted and characterized, macromodels are created that exhibit the behavior and performance of the block as implemented, and the macromodels are combined and simulated using a fast high-level simulator, such as a SPW or an AMS simulator. In practice this is done by refining the models for the blocks used during the top-down design. To reduce the chance of errors, it is best done during the mixed-level simulation procedure. Thus, the verification of a block by mixed-level simulation becomes a three step process. First the proposed block functionality is verified by including an idealized model of the block in system-level simulations. Then, the functionality of the block as implemented is verified by replacing the idealized model with the netlist of the block. This also allows the effect of the block's imperfections on the system performance to be observed. Finally, the netlist of the block is replaced by an extracted model. By comparing the results achieved from simulations that involved the netlist and extracted models, the functionality and accuracy of the extracted model can be verified. From then on, mixed-level simulations of other blocks are made more representative by using the extracted model of the block just verified rather than the idealized model. The extracted model may also be used to support reuse of the block.

3.0 Intra Flow Design Issues

The previous section described how the design process is partitioned into tasks that support the refinement of complex systems from a top level architectural concept to a working physical implementation. In this section we will analyze some design issues that cannot be addressed by enhancements to any particular point tool in a flow. Instead they require a holistic approach, where every task in the flow must participate in a comprehensive solution to the design problem. A new design methodology, better suited for more complex design objectives or for more aggressive time to market, can be made possible by the coordinated operation of all design phases.

The first consideration is the frequency range of the components of the SOC. If one or more RF components are present, simulation, verification and physical implementation must all include very different sets of models, parasitics and performance measurements.

Significant advantage in terms of cost and risk reduction for the entire design can be achieved by adopting a constraint-driven approach. However this requires transformation techniques, and every tool must understand, enforce and verify constraints.

In a similar fashion, testability considerations must be carried along the design flow and every single application must be able to understand and improve, or at least not reduce, testability of the entire chip.

Finally the communication between tools in a complex design flow, data integrity, constraint transformations and the simultaneous use of multiple models and levels of abstraction requires a strong software infrastructure with standards and interfaces to which all applications must adhere.

3.1 RF

The addition of RF to a mixed-signal chip adds considerable risk and so is done sparingly today. It is common to find the RF transceiver paths combined with a frequency synthesizer, but it is unusual to see the baseband processing or the micro-controller combined with the RF sections. This is expected to change with the development of relatively low performance RF systems such as Bluetooth and HomeRF. Here, the large volumes and low costs make a single chip implementation compelling, while the low performance requirements makes it feasible. Once success is achieved here, higher performance systems such as PCS and 3G phones are expected to be implemented on a single chip. The wireless market will be an important technology driver for MS-SOCs, and of course, including the RF sections is crucial.

There are several aspects of RF that make this a challenge. First and foremost, RF circuits operate at very high frequencies, typically between 1-5GHz. Wires that carry RF signals must be short and carefully placed to avoid interference. Floorplanning, layout, and packaging must take this into account. Accurate models are needed for the active devices, the interconnect, the package, and passive components, both on and off chip. For example, spiral inductors are used on chip and ceramic or SAW resonators are used off chip. Often exotic process are used, such as SiGe or SOI, which affects both the active and passive models. Links to field solvers and the ability to read in files of S-parameters is necessary to assure adequate verification.

Another important challenge is that RF circuits can be sensitive to interference from signals generated in the digital portion of the circuit. Signals at the input of a receiver can be as small as 1μV. Any signals that couple into the front-end of a receiver through the substrate, supplies, interconnect, or package degrades its sensitivity. The ability to accurately model these portions of the circuit and predict coupling is important.

A third challenge is that in the RF section of a transceiver, the information signal is present as a relatively low frequency modulation on a high frequency carrier. Simulating these circuits is expensive because the high frequency carrier necessitates a small time step while the low frequency modulation requires a long simulation interval. RF simulators provide special analyses that are designed to efficiently simulate these circuits, but they are incapable of including the non-RF sections [30]. One possible solution to these problems is to use the RF simulator to extract macro-models of the RF blocks that can be efficiently evaluated in an AMS simulator [40,41, 42].

3.2 Constraint Management

Especially in the design of an SOC, several levels of abstraction are used in different phases and using different models. The formulation of constraints, their management [34] along every phase in the design, their validation, verification and enforcement are

extremely critical to the consistency of the design flow. Furthermore, the design of analog and M/S systems is a process of progressive constraint refinement, where data tolerances and their level of confidence change at every step.

Physical constraints apply to the physical entities used to implement the layout. Examples are distances, area and aspect ratio, alignment between instances etc. Some commercial tools used for physical implementation such as IC-Craftsman [14] have achieved good results in enforcing physical constraints within the context of their specific application. Academic research has also devoted considerable attention to physical constraints, especially for analog design applications [5,32]. Some physical constraints such as distances, are routinely used to enforce timing and cross-talk specifications during placement and routing.

Electrical constraints apply to specific signals in the circuit. Hence, these constraints require an RTL or schematic level representation of the circuit where nets and devices are identifiable. Examples are timing, parasitics, IR drop, crosstalk noise, substrate coupling noise and electro-migration. Because of their extraordinary importance in the design of digital circuit, timing constraints need to be handled by all synthesis and physical tools. As mentioned above, special transformations into physical constraints such as net length or spacing between devices have been commonly adopted by physical tools. In the case of more complex constraints, analysis and design tools might need to be entirely redesigned to properly take them into account. An example is the case of power and ground routing with mixed analog and digital supply lines [46,47].

Finally, *design constraints* are used to characterize the behavior of individual components in terms of their I/O signals and performance. Examples are throughput, slew rate, bandwidth, gain, phase margin, power dissipation, jitter, etc. These can be specified on a circuit characterized by a model at any level of abstraction, from behavioral to physical. With complex AMS chips, design constraints might include specifications on sophisticated measurements such as distortion, noise and frequency response.

So far design constraints have not been handled adequately by commercial applications. The main reason is that their enforcement is usually impractical, since they require a transformation into a set of electrical or physical constraints in order to be handled by automatic applications. Another reason is the lack of a standard to represent these measurements and their constraints, consistent with the high-level behavioral modeling language. Such a standard should provide a description of the dependencies between electrical and design constraints when such transformation is actually performed.

A constraint management system therefore must have the following characteristics:

- It must be able to handle constraints of different types (design, electrical and physical) in a consistent way with a language applicable to all relevant description models.
- It must provide a way to facilitate transformations [31], which can be fairly complex especially for mixed-signal applications where the behavioral description of blocks might be quite abstract from the actual implementation. This includes mapping of digital-to-analog and vice versa, as well as generation of noise constraints from coupling between interconnections or through the substrate. It also includes use of behavioral and stochastic models to generate electric constraints for design centering.

- It must be able to provide a consistency check to validate constraints and detect infeasible specifications and over-constraints as early as possible in order to reduce the number of design iterations.
- Constraints must be verifiable. That means that analysis and verification tools must be able to access the definition of measurements and evaluate the corresponding performance functions using the appropriate models.

3.3 AMS Test

Generally, the last thing done for a design before it is sent to manufacturing is test program development. Verifying the test program involves running it on a working model of the chip, which is only available late in the design process. This is costly in two important ways. First, mixed-signal testers are very expensive, and test development can tie up these machines for long periods of time. Second, starting test development at the end of the design process greatly prolongs the time-to-market. If instead of running the test program on an actual chip, it can be run on a simulated version of the chip, then it is possible to address both of these issues [25].

If a top-down design methodology is used, then a system-level model of the chip exists early in the design process. This system-level model can be used during the development of the test program. Thus, the test engineers can become involved with the project much earlier, and like the block designers, are given a working virtual prototype of the chip in the form of a system-level model [10,11]. This improves communication between the test and design engineers, acts to greatly reduce the cost of test development, allows the test programs to be more thoroughly verified, and permits the test programs to be developed concurrently with the chip. All of which helps to insure that the test program is available as soon as the chip is ready to be manufactured. In addition, involving the test engineers while the design is ongoing allows fault simulation and design for test to be attempted [21].

Commercial tools are available that allow test development on virtual prototypes of the chip, but they do not as yet support Verilog-AMS or VHDL-AMS [50,55].

D. Infrastructure

The AMS-SOC design stages we have described are frequently addressed by specific tools, or mini flows, in isolation by existing EDA vendors. This is not surprising as most vendors have a rather small subset of the tools required in a complex AMS-SOC solution flow. This correspondingly restricts how much of the problem they are able to address. Without access to the internals of the tools within the flow, and without co-operation between vendors, problems cannot be addressed where they are best addressed. This leads to a patched together rats nest of tools, which can be, with a lot of wasted time, manual user intervention, and design iterations, used to create chips that eventually work. Such patching together of tools can never succeed in creating an efficient design environment capable of the fast time to market that is needed in today's AMS-SOC market.

Further, many of the tools in use today were not designed for the complexities and sizes implicit in AMS-SOCs. This manifests itself both in the analog and the digital design tools from front end through physical realization. On the analog side, most tools still target traditional transistor-based bottom-up design methodologies. The capacity of such

capture and analysis tools is inherently limited. Further, the physical realization of such designs is a largely manual process. On the digital side, the tools currently do not take sufficient account of the physical affects during the logical and planning design phases. This results in designs that cannot meet the constraints when physically realized, and thereby require costly design rework (silicon iterations). But worse than these specific limitations that are being addressed at a localized level is the interaction of the digital and analog design processes. Not only do the digital and analog design tools tend to be targeted only to their specific design methodologies, they frequently do not take into account the effects of their counterparts. Even the communication of these tools between the digital and analog domain tends to be in different forms than the other expects, thereby making it difficult, or impossible, to create an efficient design flow that ensures data integrity.

To create a truly efficient design environment for AMS-SOC design, we need to start from scratch. First, the AMS-SOC design methodology needs to be defined. Given that methodology, a design flow can be specified. This flow will then clearly dictate the necessary tools, design representations and data formats that are needed to convey all needed data both within a design stage, and across design stages. Such a definition is the contract, or infrastructure, to which all tools must conform.

Given this infrastructure it then becomes possible to design tools that will not only have the necessary functionality, but will by definition be plug-and-play in an efficient solution. By thus restricting the data locations that all tools must both read and write data to, as well as the allowed formats, it becomes possible to insert tools as needed into the flow without requiring a redesign of other components. It also becomes possible to create utilities that perform design integrity checks. With a restricted set of formats dictated by the needed abstraction levels, as opposed to the eccentricities and whims of a tool designer, the types of checks needed is greatly reduced and confined to what is required by the design methodology.

Perhaps the most import of these formats will be the MS-HDLs. They are expected to be used as a common language for representing the design and will be understood by most tools, even those from competing vendors. As such, tools other than simulators are expected to be extended to support one or both of the MS-HDLs. The MS-HDLs are also open standard languages, which means there will be greater willingness by the design and EDA communities to invest in developing model libraries and support tools for these languages. MS-HDLs are also likely to develop into a medium of exchange between block authors and block integrators.

3.5 Mixed-Signal Hardware Description Languages

Both Verilog-AMS and VHDL-AMS have been defined and simulators that support these languages are becoming available. These languages are expected to have a big impact on the design of mixed-signal systems because they provide a single language and a single simulator that are shared between analog and digital designers. It will be much easier to provide a single design flow that naturally supports analog, digital and mixed-signal blocks, making it simpler for these designers to work together. It also becomes substantially more straight-forward to write behavioral models for mixed-signal blocks. Finally, the AMS languages bring strong event-driven capabilities to analog simulation, allowing analog event-driven models to be written that perform with the speed and capacity inherited from the digital engines.

It is important to recognize that the AMS languages are primarily used for verification. Unlike the digital languages, the AMS languages will not be used for synthesis in the foreseeable future because the only synthesis that is available for analog circuits is very narrowly focused.

3.5.1 Verilog-AMS

Verilog-A is an analog hardware description language patterned after Verilog-HDL [19]. Verilog-AMS combines Verilog-HDL and Verilog-A into a MS-HDL that is a super-set of both seed languages [51]. Verilog-HDL provides event-driven modeling constructs, and Verilog-A provides continuous-time modeling constructs. By combining Verilog-HDL and Verilog-A it becomes possible to easily write efficient mixed-signal behavioral models. Verilog-AMS also provides automatic interface element insertion so that analog and digital models can be directly interconnected even if their terminal / port types do not match. It also provides support for real-valued event-driven nets and back annotating interconnect parasitics.

A commercial version of Verilog-AMS that also supports VHDL is expected soon from Cadence Design Systems.

3.5.2 VHDL-AMS

VHDL-AMS [6,22,52] adds continuous time modeling constructs to the VHDL event-driven modeling language [20]. Like Verilog-AMS, mixed-signal behavioral models can be directly written in VHDL-AMS. Unlike with Verilog, there is no analog-only subset.

VHDL-AMS inherits support for configurations and abstract data types from VHDL, which are very useful for top-down design. However, it also inherits the strongly typed nature of VHDL, which is a serious issue for mixed-signal designs. Within VHDL-AMS you are not allowed to directly interconnect digital and analog ports, and there is no support for automatic interface element insertion built-in to the language. In fact, you are not even allowed to directly connect ports from an abstract analog model (a signal flow port) to a port from a low-level analog model (a conservative port). This makes it difficult to support mixed-level simulation. These deficiencies have to be overcome by a simulation environment, making VHDL-AMS much more dependent on its environment. This should slow deployment of effective VHDL-AMS-based flows.

A commercial version of VHDL-AMS that also supports Verilog is available from Mentor Graphics [53]. A VHDL-AMS simulator is also expected soon from Analogy [53].

3.6 Design Reuse

The push to reduce costs for the consumer market place by increasing integration will result in larger and more complete systems on chip. Once mixed-signal circuits exceed a certain size, a full-custom design style becomes impractical. With circuits of this size, the AMS-SOCs described above become blocks that are combined with very large digital blocks such as micro-controllers to form ASIC-SOCs. In this case, the intended complexity of the interaction between mixed-signal blocks is relatively low and a top-down design style that includes the mixed-signal blocks is usually not necessary. The mixed-signal blocks can generally be designed with little interaction from the system engineer.

It is hoped that the mixed-signal blocks could be designed in advance as relatively generic components and incorporated into many designs. To support this, the mixed-sig-

nal blocks must be designed for reuse. At a minimum this implies that certain documentation be available that describes the block. Standards that specify what type of documentation is required have been set by the Virtual Socket Interface Alliance (VSIA) [54]. In addition, if the block is large it may be required to be embedded in special interface collars to make it easier to import them into an ASIC-SOC. These collars provide a standard interface and guard-banding to provide some degree of isolation from the rest of the circuit.

With the rapid changes in technology, and with the difficulty of migrating mixed-signal blocks to a new technology, it is generally not possible to reuse a single block design more than a few times. Thus, preparing a design for reuse must take significantly less effort than redesigning the block for a new application. An important task when preparing a block for reuse is generating a high-level model of the block that captures its essential behavior. This model is used to evaluate the suitability of the block for use in follow-on projects. It must capture the significant imperfections of the block, and must be generated as a bi-product of the block design with little extra effort by a person with limited modeling skills.

Even though the design community has become familiar with these issues and understands well the advantages, reuse today is still used infrequently. Organizations such as VSIA have taken on significant roles to define standards and methodologies. However more work needs to be done, especially to reduce the overhead on designers, and to define widely accepted practices for design for reuse and IP interchange.

The main areas for improvement are:

- Design methodologies that improve the chances a block can be reused such as interface-based design for digital blocks.
- Interface verification and IP qualification and certification.
- Tools that help create matching behavioral blocks for actual analog implementations.
- Formal and robust techniques to associate constraint sets to the behavioral description of mixed-signal blocks

Once blocks are designed and made available for reuse, it is also necessary for them to be easily accessible to other designers. As such, the ability to automatically generate datasheets for the blocks and publish them on the web so that they are easily searched and browsed by other designers. These datasheets should include an accurate high-level model that can be used to audition the block in the intended system.

4.0 Summary

In this paper we have analyzed the problems that must be addressed in the immediate future to handle the complexity of system on a chip designs for mixed-signal applications. Our analysis shows that in many areas improvements are required not only in the tools, but in the entire design methodology. A solution for AMS-SOC requires advanced tools, well defined flows, an infrastructure supporting design reuse and excellent communication between the interacting resources participating in the design flow. It also requires designers that are willing and able to change the way that they design. To change, they must have a broader set of skills, such as a understanding of modeling and a familiarity with MS-HDLs. Graduate and continuing education should be expanded to provide these skills.

A significant market is opening up for large mixed-signal consumer applications using SOC devices in the next few years. Some major EDA vendors are already positioning themselves to provide technology and comprehensive services in this arena. This effort will have to include not only large scale tools for specific tasks such as a mixed signal floor planning, but also a consistent representation for the characterization of mixed-signal behavioral data (and measurements) at all levels of abstraction and for constraints. Finally it will require utilities for design integrity checking, constraint validation, and manufacturing sign-off.

4.1 If You Have Questions

If you have questions about what you have just read, feel free to post them on the *Forum* section of *The Designer's Guide Community* website. Do so by going to *www.designers-guide.org/Forum*.

References

[1] M. Aktuna, R. A. Rutenbar and L. R. Carley. Device-level early floorplanning algorithms for RF circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 4, pp. 375-388, April 1999.

[2] L. R. Carley, R. A. Rutenbar. How to automate analog IC designs. *IEEE Spectrum*, vol. 25 no. 8, pp. 26-30, Aug. 1988.

[3] H. Chang, E. Charbon, U. Choudhury, A. Demir, E. Felt, E. Liu, E. Malavasi, A. Sangiovanni-Vincentelli, I. Vassiliou, *A Top-Down, Constraint-Driven Design Methodology for Analog Integrated Circuits*, Kluwer Academic Publishers, 1997.

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

[5] J. M. Cohn, D. J. Garrod, R. A. Rutenbar and L. R. Carley. *Analog Device Level Layout Automation*, Kluwer Academic Publishers, 1994.

[6] E. Christen, K. Bakalar. VHDL-AMS — a hardware description language for analog and mixed-signal applications. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, no. 10, pp. 1263 -1272, Oct. 1999.

[7] G. Debyser, G. Gielen. Efficient analog circuit synthesis with simultaneous yield and robustness optimization. *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers (ICCAD '98)*, pp. 308 -311, Nov. 1998.

[8] M. Degrauwe, et al. IDAC: An interactive design tool for analog CMOS circuits. *IEEE Journal of Solid-State Circuits*, vol.22, no.6, pp.1106-1116, Dec.1987.

[9] M. G. R. Degrauwe, B. L. A. G. Goffart, C. Meixenberger, M. L. A. Pierre, J. B. Litsios, J. Rijmenants, O. J. A. P. Nys, E. Dijkstra, B. Joss, M. K. C. M. Meyvaert, T. R. Schwartz and M. D. Pardoen. Towards an analog system design environment. *IEEE Journal of Solid-State Circuits*, vol. 24, no. 3, pp. 659-671, June 1989.

[10] C. Force, T. Austin. Testing the design: the evolution of test simulation. *International Test Conference*, Washington 1998.

[11] C. Force. Integrating design and test using new tools and techniques. *Integrated System Design*, February 1999.

[12] G. G. E. Gielen, H. C. C. Walscharts and W. M. C. Sansen. ISAAC: A symbolic simulator for analog integrated circuits. *IEEE Journal of Solid-State Circuits*, vol. 24, no. 6, pp. 1587-1597, December 1989.

[13] G. Gielen, P. Wambaq and W. Sansen. Symbolic analysis methods and applications for analog circuits: a tutorial overview. *Proceedings of the IEEE*, vol. 82, no. 2, pp.287-304, February 1994.

[14] R. Goering. Cadence ties routing to RC extraction. *Electronic Engineering Times*, September 28, 1998,

[15] R. Goering. Cadence claims synthesis coup, *Electronic Engineering Times*. July 12, 1999.

[16] R. S. Gyurcsik and J. C. Jean. A generalized approach to routing mixed analog and digital signal nets in a channel. *IEEE Journal of Solid-State Circuits*, vol. 24, no. 2, pp. 436-442, April 1989.

[17] M. D. M. Hershenson, A. Hajimiri, S. S. Mohan, S. P. Boyd and T. H. Lee. Design and optimization of LC oscillators. *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers (ICCAD '99)*, pp. 65-69, November 1999.

[18] J. Holmes, F. James, and I. Getreu. Mixed-signal modeling for ICs. Integrated System Design Magazine, June 1997.

[19] *Standard Description Language Based on the Verilog™ Hardware Description Language,* IEEE Standard 1364-1995.

[20] *VHDL Language Reference Manual,* IEEE Standard 1076-1993.

[21] *IEEE Standard for a Mixed-Signal Test Bus*. IEEE Standard 1149.4-1999.

[22] *Definitions of Analog and Mixed-Signal Extensions to IEEE Standard VHDL.* IEEE Standard 1076.1-1999.

[23] A. B. Kahng and Y. C. Pati. Subwavelength lithography and its potential impact on design and EDA. *Proceedings of the 36th Design Automation Conference (DAC '99)*, pp.799-810, June 21-25, 1999.

[24] M. Keating and P. Bricaud. *Reuse Methodology Manual for System-on-a-Chip Designs*. Kluwer Academic Publishers, 1998.

[25] N. Khouzam. Simulating mixed-signal tests to reduce time-to-market. Integrated System Design, April 1997.

[26] A. Knight. *Basics of Matlab and Beyond*. CRC Press, 1999.

[27] A. Konczykowska and M. Bon. Automated design software for switched-capacitor IC's with symbolic simulator SCYMBAL. *Proceedings of the 25th Design Automation Conference (DAC '88)*, pp.363-368, 1988.

[28] H. Y. Koh, C. H. Sequin and P. R. Gray. OPASYN: A compiler for CMOS operational amplifiers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 2, pp. 113-125, February 1990.

[29] M. Krasnicki, R. Phelps, R. A. Rutenbar, L. R. Carley. MAELSTROM: efficient simulation-based synthesis for custom analog cells. *Proceedings of the 36th Design Automation Conference (DAC '99)*, pp. 945-950, June 1999.

[30] K. Kundert. Introduction to RF simulation and its application. *IEEE Journal of Solid-State Circuits*, vol. 34, no. 9, pp. 1298-1319, Sept. 1999. Available from *www.designers-guide.org*.

[31] E. Malavasi and E. Charbon. Constraint transformation for IC physical design. *IEEE Transactions on Semiconductor Manufacturing,* vol. 12, no. 4, pp. 386-395, Nov. 1999.

[32] E. Malavasi, E. Charbon. E. Felt and A. Sangiovanni-Vincentelli. Automation of IC layout with analog constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 8, pp. 923-942, Aug. 1996.

[33] E. Malavasi, D. Guilin, K. Jones and W. Kao. Layout acceleration for IC physical design. *Proc. DATE-99 (User Forum)*, pp.7-11, March 1999.

[34] E. Malavasi and W. H. Kao. Constraint-driven physical design issues for a mixed-signal flow. *Proc. ED&TC-97 (User Forum)*, pp.63-67, March 1997.

[35] W. Nye, D. C. Riley, A. Sangiovanni-Vincentelli and A. L. Tits. DELIGHT-SPICE: An optimization-based system for the design of integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 4, pp. 501-519, April 1988.

[36] E.S. Ochotta, R.A. Rutenbar and L.R. Carley. ASTRX/OBLX: tools for rapid synthesis of high-performance analog circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, n. 3, March 1996.

[37] S. Ohr. Cell builder tool anticipates analog synthesis. *Electronic Engineering Times*, 9 November 1998.

[38] S. Ohr. Pay per use op-amp synthesizer hits the net. *Electronic Engineering Times*, 10 April 2000.

[39] S. Ohr. Barcelona adds RF passives to online tool suite. *Electronic Engineering Times*, 1 May 2000.

[40] J. Phillips. Model reduction of time-varying linear systems using approximate multipoint Krylov-subspace projectors. *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers (ICCAD '98)*, pp. 96-102, Nov. 1998.

[41] J. R. Phillips. Automated extraction of nonlinear circuit macromodels. *Proceedings of the 2000 IEEE Custom Integrated Circuits Conference (CICC '00)*, May 2000.

[42] J. Roychowdhury. Reduced-order modelling of linear time-varying systems. *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers (ICCAD '98)*, pp. 92-95, Nov. 1998.

[43] R. A. Rutenbar. Analog design automation: Where are we? Where are we going? *Proceedings of the 1993 IEEE Custom Integrated Circuits Conference (CICC '93)*, pp. 13.1.1-13.1.7, May 1993.

[44] R.A.Rutenbar and J.M.Cohn. Layout tools for analog ICs and mixed-signal SoCs: a survey. *Proc. Intl. Symposium on Physical Design*, pp.76-82, April 2000.

[45] S. J. Seda, M. G. R. Degrauwe and W. Fichtner. A symbolic analysis tool for analog circuit design automation. *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers (ICCAD '88)*, pp.488-491, November 1988.

[46] B. R. Stanisic, N. K. Verghese, R. A. Rutenbar, L. R. Carley and D. J. Allstot. Addressing noise decoupling in mixed signal ICs: simulation and power distribution synthesis. *IEEE Journal of Solid-State Circuits*, vol. 29, no. 3, pp. 226-238, March 1994.

[47] B. R. Stanisic, R. A. Rutenbar, and L. R. Carley. Addressing noise decoupling in mixed-signal IC's: power distribution design and cell customization. *IEEE Journal of Solid-State Circuits,* vol. 30, no. 3, pp. 321 -326, March 1995.

[48] *Signal-Processing Worksystem User's Guide*. Cadence Design Systems, San Jose, CA.

[49] B. R. Stanisic, R. A. Rutenbar and L. R. Carley. *Synthesis of Power Distribution to Manage Signal Integrity in M/S ICs*. Kluwer Academic Publishers, 1996.

[50] SpectreVX and SaberVX virtual test environments, *www.teradyne.com*.

[51] *Verilog-AMS Language Reference Manual: Analog & Mixed-Signal Extensions to Verilog HDL*, version 2.1. Accellera, January 20, 2003. Available from *www.accellera.com*. An abridged version is available from *www.verilog-ams.com* or *www.designers-guide.org/VerilogAMS*.

[52] VHDL-AMS, *http://www.vhdl.org/analog*.

[53] VHDL-AMS simulators, *http://www.vhdl-ams.com*.

[54] Virtual Socket Interface Alliance Official Web Page, *www.vsi.org*.

[55] Dantes virtual test environment, *www.virtualtest.com*.