The IBM ASIC/SoC methodology—A recipe for first-time success

by G. W. Doerre
D. E. Lackey

This paper describes the methodology employed by the IBM Microelectronics Division for the design of its Blue Logic® application-specific integrated circuits (ASICs) and system-on-a-chip (SoC) designs. This methodology is used by both IBM ASIC and SoC designers, as well as OEM customers. A key focus of the IBM ASIC/SoC methodology, outlined in the first section of this paper, is the first-time-right methods of design and verification that maximize correct operation of the chip upon product integration. The second section of this paper describes advances in methodology that deal with the physical effects of shrinking device geometries and enable design using the performance and density capabilities available in the new technologies, and methodology advances that have improved design turnaround time (TAT) for large, complex designs. Upcoming nanometer-level technologies present new opportunities to integrate systems on a single chip, including functional components of mixed libraries and mixed analog and digital design. The final section of this paper outlines strategies that are enabling SoC design at these levels.

Introduction

While the device dimensions and structures, chip capacities, performance levels, and diversity and range of intellectual property (IP) in the VLSI and ASIC/SoC industries are most frequently described by silicon product vendors, the methods and execution time of the underlying design and integration processes are also critical factors in the success of product designers.

Starting more than thirty years ago, based on the need to integrate multi-million-gate computer systems from thousands of hundred-gate chip designs, the modern ASIC industry is on the threshold of 100-million-gate chip design capability. Now many processors can be integrated onto a single system-on-a-chip (SoC). Although multi-million-gate ASIC and SoC designs are now routinely manufactured, designing them correctly and producing them on time, and in volume, with adequate quality/reliability levels, all involve the *methodology* of design.

ASIC/SoC methodologies are needed that offer designers the integration of systems with a complete range of reusable digital and analog functions, and ways to integrate them onto a single chip [1]. Electronic design automation (EDA) companies are focusing as much on tool flow and integration as on the development of traditional standalone tools in order to relieve the complexity burden of ASIC/SoC design [2]. Finally, a

0018-8646/02/$5.00 © 2002 IBM

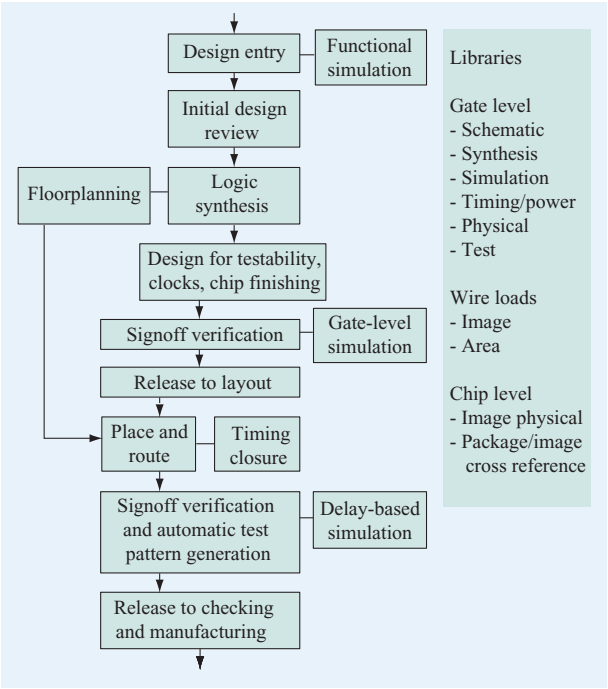

Figure 1

Early 1990s ASIC methodology flow.

range of specialized third-party products and services, from standard libraries to unique IP and chip-level design, are now emerging to cover niches in the design process and functional product spectra [3–5].

This paper describes the development of a single methodology that the IBM Microelectronics Division has used to provide a broad range of capabilities in hundreds of ASIC/SoC designs.

The fundamental ASIC methodology

In the early 1990s, IBM became a commercial provider of VLSI technology, ASIC methodology, and services. This transition from internal supplier to industry provider generated two fundamental questions regarding IBM design tool strategy:

1. Which EDA tools (third-party vendor or IBM tools) would be supported for ASIC design?
2. What should the methodology flow include (in particular, front-end design, design signoff, manufacturing test, and chip layout)?

To allow external ASIC customers to easily adapt their chip designs and logic design methodologies to the IBM offering, IBM provided support for front-end design tools from leading third-party tool vendors. In particular, IBM worked with logic synthesis vendors to ensure that key optimization capabilities were effectively exploited by the IBM ASIC libraries. Comprehensive VHDL [6] and Verilog [7] libraries were developed to support use by customers of a wide variety of third-party high-performance logic simulation tools. Other vendor offerings, including tools for logic entry, power analysis and estimation, and formal verification, were similarly supported in the IBM methodology flow. IBM leveraged its internal design tools in three key areas: static timing analysis, design for testability (DFT), and chip physical layout.

The IBM Microelectronics Division developed its Blue Logic* ASIC design methodology, which enabled hundreds of ASICs and SoCs to be designed and operate successfully in the customers' products. **Figure 1** depicts the sequence of steps executed by the designer within a set of IBM and third-party design tools that constitute the methodology described in this section: design entry, design reviews and checkpoints, simulation and other checking and verification, logic synthesis, physical floorplanning, DFT and automatic test-pattern generation (ATPG), clock design, physical layout (place-and-route and timing closure), and design release.

Register-transfer-level design and logic synthesis

Register-transfer-level (RTL) design descriptions were a fundamental entry point for ASIC design. The growing number of circuits in a typical ASIC design, and the corresponding time required to design products of increasingly functional complexity, required design representations at a higher level of abstraction than a gate-level design. RTL design languages, such as VHDL and Verilog, provided a way to describe sequential and combinational functions at the higher behavioral level. The focus on RTL capability included methods to incorporate macros such as arrays and embedded intellectual property (IP) in the form of core functions in either a hard core (full layout) or soft core (pre-layout gate-level logic description). IBM created software to generate functional interfaces with macros for rapid integration in the RTL chip design, and to automate the DFT signal connections to the macros [8].

Adoption of RTL methods led to the widespread use of logic synthesis tools that map RTL descriptions to gate-level implementations and optimize the timing performance of the implementation. ASIC designers used third-party logic synthesis tools to map their RTL to gates based on gate representations in circuit libraries, and to optimize the resulting implementation for timing and area. An important criterion for synthesis tools and for the modeling of circuits for synthesis tools was maintaining close correlation between the timing data in the synthesis

models and the timing calculated with static timing analysis sign-off tools.

In the early 1990s, statistical wire-load models (WLMs) were developed to provide estimates of interconnect delay, based on an estimated average distance between connecting gates. The values provided in the WLMs were based on die size, assuming that connecting gates could appear anywhere on the chip. As technology products offered increasing circuit densities, the contribution of interconnect delay to overall logic path delay became more significant. Beginning in 1995 with the CMOS 5L technology product, IBM created area-based WLMs that allowed reduced interconnect delay estimates based on a more aggressive assumption that the connecting gates would be constrained within a corresponding chip area [9]. Increased interconnect delay led to the partitioning of chip-level logic designs and their assignment to physical areas on the chip. Floorplanning tools were increasingly used for planning the logic partitions, creating regions on the chip for the partitions, and managing the WLMs to be used by synthesis tools.

Static timing analysis and timing sign-off

By the early 1990s, static timing analysis was well established within IBM as a highly productive method of evaluating the performance of digital logic and interconnect wiring between logic gates. Static timing tools measure delays of all combinational logic paths and report whether each path has met or failed to meet the timing requirements established by the product designer, and by how much. A reliable timing "signoff" can then be performed at each step of the design cycle: The results reported by the static timing tool are audited by the product designer and the ASIC vendor to determine whether the design can pass to the next step in the design cycle, and ultimately whether the design can proceed to manufacturing. Static timing analysis is efficient and exhaustive for timing path coverage, and thus more effective than the alternative offered by delay-based gate-level simulation [10].

Race-free full-scan design for testability

The logic synthesis flow and the IBM ASIC DFT methods have enabled designers to largely ignore hardware test issues when designing the functional logic. While providing this functional design flexibility, these DFT test techniques also provide extremely high levels of stuck-fault coverage (>99% on most ASICs) using ATPG software. Through the use of these DFT methods (which include software for DFT design automation and verification) and ATPG, all patterns required for ASIC manufacturing test are generated, and the ASIC designer is freed from the need to develop test patterns.

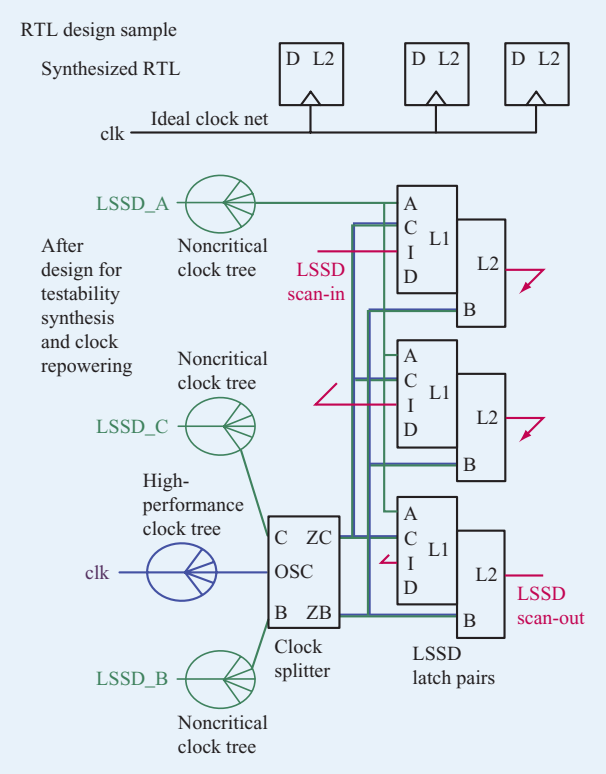

Figure 2

Automation of LSSD latch and clock mapping through the design for testability and clock synthesis flows.

The DFT methods are based upon level-sensitive scan design (LSSD) [11]. All latches are connected into shift registers known as scan chains. For test operation, LSSD provides unique control of the clocking of the two latches that comprise each register bit, as shown in **Figure 2**. For the functional operation of the product, a single system clock operates each latch pair as an edge-triggered flip-flop. Separate clocking of the LSSD latch pair in the test operation allows all scan operations and testing for stuck-faults to be performed independently of the timing behavior of the logic paths. Thus, LSSD is known as a "race-free" test capability. Tools were developed [12] to automate the insertion of DFT structures into the ASIC design. The sequence by which the DFT synthesis tools create an LSSD-compliant design from the output of logic synthesis tools, and the repowering of the system and test clocks, is depicted in Figure 2. The combination of design automation, full-scan DFT structures, and race-free test operation allows the RTL designer to largely ignore hardware test issues when designing the product function.

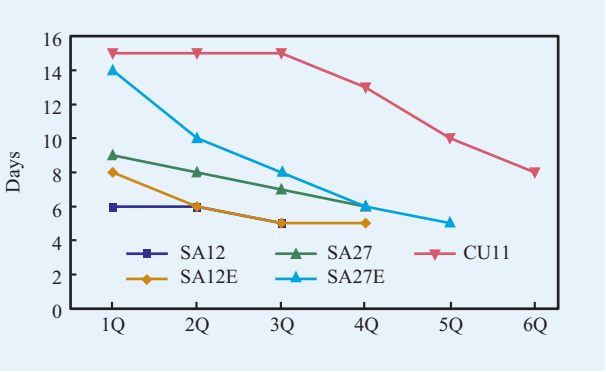

Figure 3

Turnaround time reduction for final ASIC checks.

Physical design and correct-by-construction images

An ASIC design is composed of the following basic types of physical components with varying design objectives:

- *Circuits used as logic gates and macros.* Placement and interconnection involving these elements and their interconnection is focused largely on logic performance and avoidance of wiring congestion.
- *Clock drivers.* Placement and interconnection of these elements is focused on minimizing clock skew and meeting targets for latency and switching transition time.
- *Circuits used for signal input/output (I/O) and electrostatic discharge (ESD).* Placement of these elements is both determined and restricted by factors that include power distribution, noise avoidance, and package constraints, as well as the logic design itself.
- *Wiring.* This is a unique pattern of interconnections among the circuits described above that fits within the metal area available (wiring resource) for interconnection on the chip.
- *Filler cells.* These provide separation between certain types of circuits and voltage biasing cells.

An ASIC chip structure is made up of these basic physical components:

- Wiring for power distribution.
- Package interconnection points for signal, test, and power-supply I/O.
- Pre-defined (legal) placement locations and restrictions for the various circuit types.

IBM has maintained competency in the placement and routing of ASIC designs. Simplistically, a placement tool optimizes the location of circuits on the die to meet the timing requirements set by the product designer [13, 14], while conforming to placement restrictions (e.g., for I/O cells) and requirements for electrostatic discharge cells, fillers for separation of some cell types, and biasing cells to satisfy electrical requirements of the technology, as well as legal placement locations. The placement tool places the circuits optimally to provide adequate space for wiring, while the routing tool provides an electrically correct and uncongested distribution of interconnect wiring while meeting the timing requirements.

The IBM Microelectronics Division has minimized the occurrence of product problems related to power distribution and electrical requirements by providing pre-defined chip architectures (images) for varying die sizes and package designs. These images include pre-designed power distribution grids, signal I/O connection points, and rows of legal placement locations for the various circuit types.

Final checking

Once an ASIC design is completed, a series of final checks are run before the design is released to manufacturing:

1. Verification checks that no changes to the original functional design have resulted from logical structural changes that occur throughout the design flow (for DFT and clock automation, and for optimization); formal logic equivalence checking [15] is used, comparing the Boolean equations of the design version prior to structural changes to the version following.
2. Logic structure checks required by the CMOS manufacturing and test equipment, estimations of noise and electromigration effects, and placement and wiring checks.
3. DRC/LVS checks to ensure that process rules were not violated during the physical design process.
4. Final static timing analysis and DFT verification.

With these checks, IBM achieved a record of first-pass success. As designs became larger, the run time for these checks had grown to almost two weeks. Through a combination of early testing, parallelization, and other innovative approaches, this turnaround time (TAT) has been reduced (see **Figure 3**).

Summary—the early 1990s

The technical and organizational capabilities that were put in place allowed IBM to quickly build a record of success in ASIC design. This success was fueled by

- First-time-right quality (built upon static timing sign-off, a comprehensive test methodology, and a comprehensive verification system).
- Large-design enablement through the capabilities and capacity of the tool set.

- Technology capabilities: circuit performance and density, I/O signal performance, and packaging.
- A full range of services for ASIC design (a front-to-back methodology and front-end through physical design services).

Building upon these capabilities, the ASIC methodology evolved to meet emerging technology challenges and industry requirements, as described in the next section.

Rapid ASIC technology growth at IBM

In 1999, six years after entering the ASIC market, IBM Microelectronics became the largest ASIC supplier. During this period of rapid growth, ASIC technology at IBM continued to progress according to Moore's law [16], which predicted a continued doubling of on-chip processing capability every eighteen months.

Shrinking device geometries required synthesis and placement capabilities that accounted for interconnect delay more accurately than WLMs. Better routing solutions were needed to avoid signal routing congestion, repower clocks at higher performance levels, and address signal integrity and noise avoidance issues. Because I/O circuits had been limited to the chip periphery, the number of I/O signals could not grow as quickly as the number of internal circuits, causing a communication bottleneck predicted by Rent's rule [17]. All of these elements emphasized the need for a hierarchical design process that managed the larger design content, maintained or reduced design turnaround time (TAT), and completely automated DFT.

Timing closure

Historically, ASIC design methodologies separated logical and physical design. In 0.25-μm and earlier process technologies, gate delay dominated interconnect delay. Therefore, logic synthesis tools could use a rough approximation of interconnect delay or ignore it completely. Similarly, physical design focused on wirability and not on timing.

With smaller geometries, interconnect delay became a larger part of the overall delay and had to be factored in for both synthesis and physical layout. This was partly addressed with statistical wire-load models (WLMs) for synthesis. The WLM-based timing closure design flow contained convergence loops of the following three types (**Figure 4**):

1. Iteration between floorplanning and synthesis.
2. Iteration of timing and wirability fixes in layout.
3. Iteration between layout and floorplanning.

Even with statistical WLMs, logical/physical timing correlation continued to degrade. Placement-based synthesis [18], which calculates interconnect delays based on an actual design placement, was developed in the late 1990s to address this. Commercial placement-based-synthesis tools are now generally available [19, 20], along with the IBM Placement-Driven Synthesis (PDS) tool [21]. A placement-based-synthesis methodology contains three components:

1. *Synthesis-determined placement.* A placement engine embedded in the synthesis tool allows synthesis to determine gate placements. Synthesis drives the gate-placement assignments together with the logical implementation to optimize timing while achieving a wirable design.
2. *Understanding of the placement by synthesis.* The synthesis tool estimates a wire route based on gate placement, and uses this wire route to estimate interconnect delay. The wirability of the placement is also determined from the estimated route.
3. *Synthesis-based placement optimization.* During final layout, placement-based synthesis is used for late timing correction, using an existing placement and

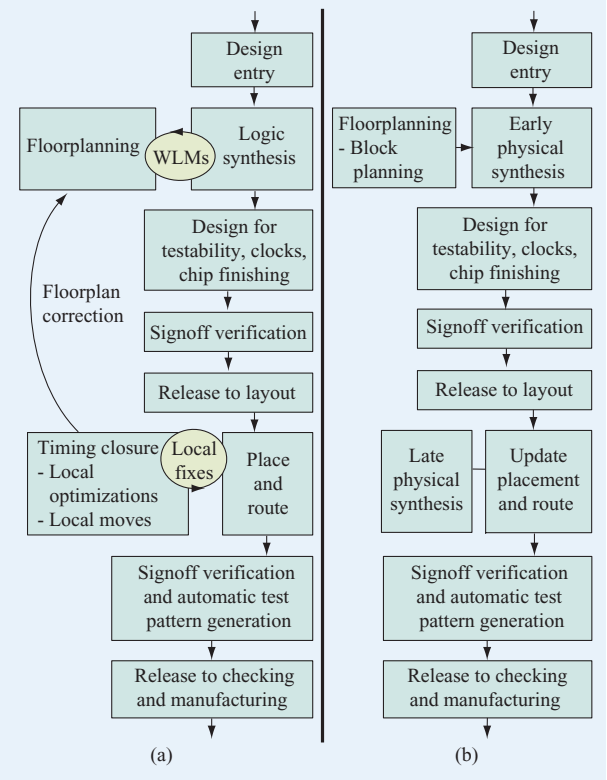

Figure 4

Timing closure improvements over traditional methodology: (a) Traditional iterative timing closure flow; (b) improved non-iterative timing closure flow with physical synthesis.

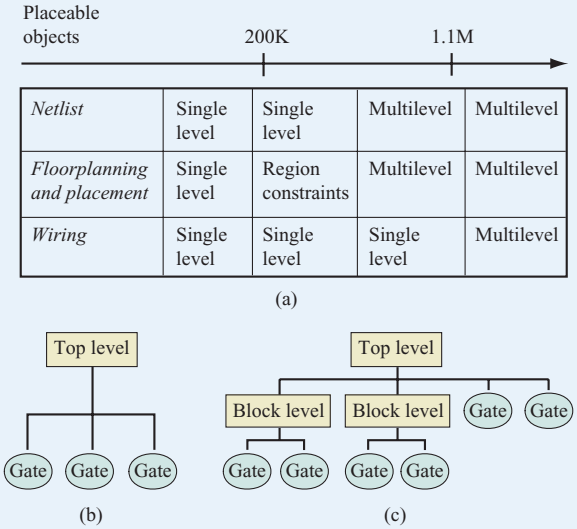

Figure 5

Flat vs. hierarchical design tradeoffs: (a) Flat design; (b) hierarchical design.

routing. Logic and placement changes are more restricted in this phase.

The IBM PDS tool has been enhanced with timing-driven global placement techniques, quadratic placement methods, design partitioning using quadrisection techniques, placement-aware methods of buffering and repowering, and control of clock delay [21–23].

Area array I/O—Defeating Rent's rule

Until the late 1990s, as the number of logic gates on a chip increased, the number of required signal I/Os increased, but at a slower rate, generally following Rent's rule [11]. IBM addressed this impending bottleneck with an area array I/O architecture [24], which covered the entire chip with a uniformly spaced I/O pin array, more than tripling the number of signal I/Os. However, as long as pin spacing remains constant, the problem of increasing gate density remains, until a point is reached at which Rent's rule no longer holds. When integration levels increase to the point at which major logic functions are connected by buses, the number of signal I/Os needed will drop substantially, and then remain almost constant, even as gate count increases. This is one of the key inflection points between the ASIC and the system-on-a-chip (SoC) eras.

Hierarchical vs. flat physical design tradeoffs

Today, for designs with fewer than 200K circuits, it is quicker and easier to use a flat design methodology, because the entire chip can be processed within a few hours, unconstrained by either hierarchical design boundaries or the need to share routing resources. For designs with 1M circuits or more, hierarchical layout becomes a necessity: Tool capacity (memory requirements and run time) make flat layout prohibitive.

Between these two extremes, IBM developed the concept of *flexible hierarchy*. In one method, a flat layout is subdivided into *region constraints*, and each region is driven with a targeted set of placement objectives. Regions are a soft boundary, however, allowing top-level objects to be placed within the region, or allowing regions to overlap. In another method, particularly important when placement-based synthesis tools are used, the hierarchy of the design is used during the placement operation, which provides parallel processing capabilities late in the design cycle. Also, like region constraints, hierarchical placement allows a targeted set of placement objectives for each hierarchical block. For both of these methods, the design is flattened before routing, which has the advantage of providing the router an unconstrained problem: All metal wiring layers are available to any interconnect route. In a full hierarchical design, metal layers within the area of a block must be shared between the block and the top level. **Figure 5** depicts key considerations for flat and hierarchical design methods.

Routing improvements

Routing, which had been constrained to a simple grid, has recently become more complicated. Wider wires are being used for long wiring runs, and in wire delay/load balancing. Typically, these types of fatter wires are offered at multiple widths of the wiring grid, but wiring tools allowing variable width and spacing are now becoming available. As interconnect-based *RC* delays increase, net segments become more isolated from their drivers and more susceptible to coupling from nearest neighbors, which can introduce timing jitter. This jitter has to be taken into account, and mutually disruptive signals may have to be physically separated.

As a further complication, several types of routing must be done concurrently:

- *I/O routing* – Routing between an I/O pin, its associated buffer, and the internal chip logic.
- *Clock routing* – The high-fan-out, high-load interconnect of on-chip clocks, with precise skew management.
- *Power routing* – Routing that ensures acceptable static IR drop and contains voltage or current spikes during large-scale switching.
- *Critical signal routing* – Unique balancing requirements, including full balanced differential routing.

- *Support routing* – High-fan-out nets supporting nonfunctional operations (e.g., testing), which may or may not have an at-speed performance requirement.

The most recent generation of IBM routing tools was developed jointly with the University of Bonn [22, 23].

Dynamic image generation (image-on-the-fly)

With advances in ASIC process technology, pre-designed "off the shelf" chip background images became impractical, for several reasons:

- *Large number of permutations.* Each chip size, I/O interconnect scheme, package type, and number of wiring levels requires a unique image. As the number of permutations increases, pre-design/analysis becomes prohibitive.
- *Performance requirements and deep submicron effects.* Wiring pitches are smaller, increasing coupling effects, while voltage-supply levels/margins are decreasing.
- *Physically large/constrained cores and IP.* Arrays and specialized logic may require different line pitches and rules, or even different power supplies.

In the future, image designs will be created dynamically, or "on the fly," and the chip size, I/O type for each legal pin location, and localized wiring and placement terrains for different chip regions will be specified separately. This is another inflection point in the transition from ASICs to SoCs.

DFT automation

Before 1999, IBM Blue Logic ASICs offered insertion of scan chains into the logical description (netlist) of the chip. This included mapping of flip-flops to LSSD latches, connecting the latches into scan chains, balancing the number of bits in each scan chain to maximize test efficiency, and connecting test clocks to the latches. However, many aspects of the DFT logic remained application-specific, including chip-level connections of scan chains and test clocks to I/O cells, the logic required to share functional I/O signals and test I/O signals on the same chip pin, and boundary scan structures.

Additionally, many ASIC customers require compliance to the IEEE 1149.1 test standard [25], a method to control multiple chips in a board or product for chip-interconnect testing and other customer-defined DFT methods unique to their products or corporate test strategies.

To address automation of chip-level and custom test logic, IBM developed the IOSpecDFT tool [26]. Like many tools used in the industry, IOSpecDFT provides an automated insertion capability of I/Os and IEEE 1149.1-compliant structures. But IOSpecDFT goes further by providing a means to easily describe customer-defined test logic and inserting all of the resulting chip-level test logic.

Standard ASIC I/O port requirements described in an IOSpecList file include I/O cell, system logic function of the I/O, a test logic function, and product test logic requirements. Custom requirements that cannot be inferred automatically from the IOSpecList can be described in a connections file. These include custom boundary connections, special system test controllers (such as array BIST, logic BIST, or scan debug), or any other desired by the customer at the chip top level. IOSpecDFT creates a chip-level logic implementation based on the requirements specified in these two files.

The capabilities offered by IOSpecDFT, when coupled with automated scan-chain insertion and automated connections of macro test signals, have provided ASIC customers with an integrated and automated method for implementing the customer's in-product test and the IBM manufacturing test requirements.

Scaling, TAT, and infrastructure

An ASIC methodology may contain more than 150 steps, as well as complex iterative loops. To manage TAT for these iterations, customers are demanding that design methodologies be fully hierarchical—they want to be able to design parts of their chip separately from one another in the following ways:

- *Verification and engineering change order (ECO) localization.* Verification of complex logic now often requires as much, if not more, effort than the logic design itself. Once customers have verified a section of the design, they do not want to give up or risk that integrity.
- *Reuse.* Customers want to be able to reuse their logic. They may want to migrate a design from an older to a newer ASIC technology, and make functional changes while doing so. They may want to preserve a placement, a topology, and/or a level of performance.
- *Design concurrency.* Customers manage TAT by doing sections of the design in parallel. This may require parts of the design to be integrated for simulation.

To manage this, IBM Microelectronics has developed a design infrastructure called eDesign, and a methodology integration platform called TheGuide (**Figure 6**), that will be used together to support collaborative hierarchical design, making geographic separation, tool and library versioning, and different levels of design completion as transparent to the designer as possible.

Future ASIC/SoC technology in IBM

Technology and product offering sets are being expanded to include low-power and multiple-voltage products, reconfigurable logic, custom design capability, and

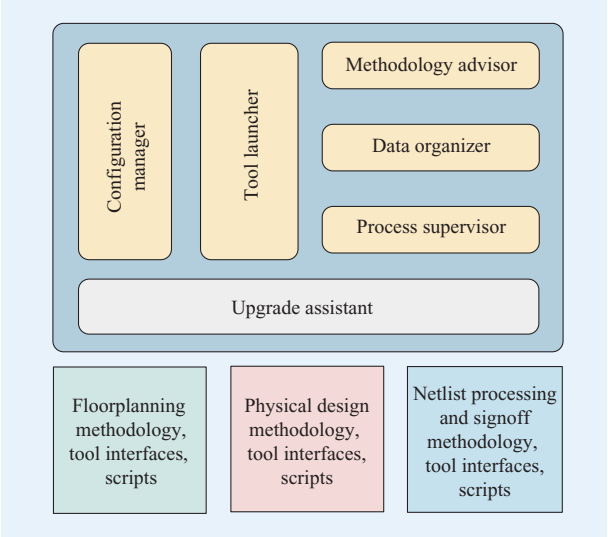

Figure 6

Architecture of TheGuide.

analog/mixed-signal designs. Each of these offerings requires a well-balanced coupling of technology capabilities and design methodology. Massive growth in the amount of logic available on a chip must be addressed, with a significant reduction in the test volume required for a given design content. Flexible and lower-cost methods must be applied to the problem of alternating-current (ac) or time-based testing.

Tool interoperability

An integrated design flow has been fundamental to IBM ASIC success. This flow includes tools that are used in a standalone manner, such as logic synthesis, static timing analysis and signoff, timing-driven and congestion-driven placement, global and local routing, layout optimization, editing tools for logic and layout, and post-layout extraction tools. What allows these tools to be used standalone or integrated is their underlying integrated control structure and design data model, and common technology libraries (see **Figure 7**).

Common to all tools is a set of application program interfaces (APIs) [27], which provide access and modification to the design data (logical and physical design content, properties, and constraints). The design data resides in a common data model which is resident in workstation or distributed memory and has an accompanying file format. Built upon the common data model and accessed by the APIs is a set of subsystems (for example, timing/electrical and wiring) that are accessed by the APIs and provide consistency across all applications for timing calculations, constraint semantics, wire-based calculations and estimates, etc.

Although industry-standard file formats, such as Verilog and VHDL for logic and PDEF [28] for layout data can improve communication between disjoint tools, these file-based methods cannot compare to the incremental and tightly coupled capabilities of tools built upon an interoperable paradigm. What is needed is industry-level standardization of libraries, data models, and APIs. The IEEE 1481 timing and power standard (modeling languages and APIs) that was led by IBM has been adopted by several EDA vendors and silicon vendors [28]. A fully interoperable set of tool APIs and data models is necessary to exploit the capabilities available across the many EDA tool vendors.

Hierarchical design for systems-on-a-chip

As system-on-a-chip (SoC) designs become more complex, the limitations and homogeneity of a flat approach become more problematic:

- *Locality.* Customers generally favor hierarchical logic design because, at an unplaced netlist level, hierarchy imposes no design constraints. A hierarchical approach allows the design to be partitioned so that several designers can work in parallel, once the logical interfaces between their parts of the design are fixed.
- *Tool performance.* With 64-bit addressing, tools may be run flat for very large designs, but run times can extend to several days. Parallelism and design partitioning can reduce these times substantially.
- *Reuse.* Customers may want to reuse part of an earlier design—at the RT level, the gate-level netlist, or the placed/routed/timed design. They may want to use the design as is, or make changes. Reuse is more difficult if the designer cannot explicitly isolate the reused design from the new work.
- *Diversity.* An SoC may have logic elements that must be physically optimized, including unique gate-level design and wiring terrains. Annotating a design so that the physical design tools can track and regionalize this diversity becomes almost impossible in a flat methodology.

Design planning and RTL signoff

Traditional chip design methodologies have been serial in nature [29], as shown in **Figure 8(a)**. In the traditional design flow, much of the iteration is caused by functional changes to the design during the ASIC design process. The serial nature of the methodology flows can be reduced by separating these two design step types, as shown in **Figure 8(b)**. Steps involving design IP should be moved to the front of the overall design flow. Steps involving design processing and optimization should be deferred as long as possible. The objective is to achieve

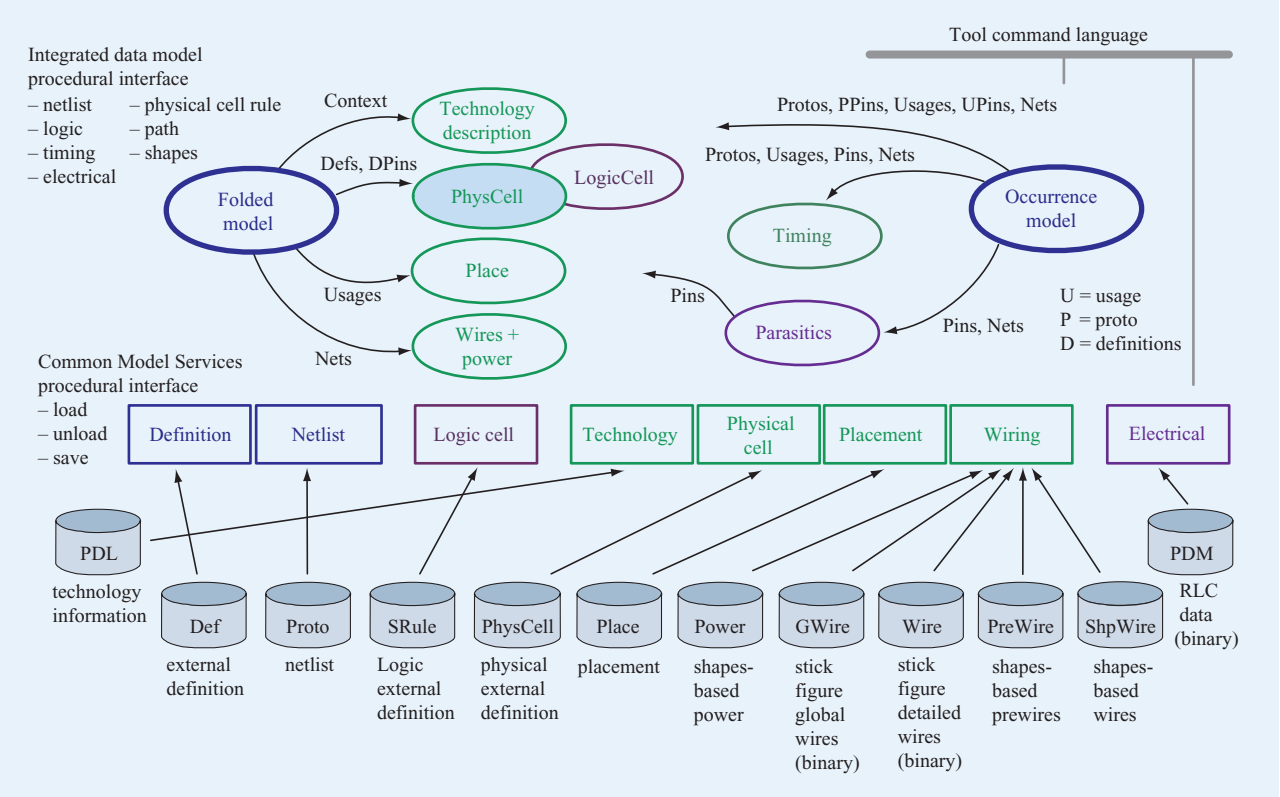

Figure 7

Integrated data model.

design (i.e., timing, power, wirability, noise) closure in a single pass. To do this, chip design planning flows must provide the following:

- RTL analysis and partitioning
  - RTL and technology characterization to estimate physical size, timing, and power.
  - Regrouping the logical hierarchy into physical blocks to optimize the interconnecting signals between blocks.
  - Block-level physical sizing and time-budgeting based upon chip-level constraints.
- High-level floorplanning
  - Optimizing I/O locations for package requirements on the basis of block, pin, and latch placements. I/O cells carry special placement restrictions due to frequency, switching, and voltage requirements of chip-interconnect signals and their susceptibility to noise. The floorplanning tool must optimize placement within these restrictions.
  - Simultaneous consideration of key placement constraints for the top level and for each block. These include block pin locations, the size, shape, and aspect ratio of each block, and the location and orientation of each block at the top level.
- Estimation/metrics extraction and design data generation
  - Timing.
  - Wirability estimates (congestion).
  - Static and dynamic power consumption.
  - Testability (scan-based coverage).
  - Partition data: block shapes, I/O and block pin placements, macro placements.
  - Synthesis data: timing budgets and wire loads.

Low-power design and voltage islands

Power-managed design is becoming a pervasive need for even the highest-performance designs, which may not be allowed to use any more power than the system they replace. Trading off power for performance must be accomplished differently for each design because of the different market requirements [30]. For some designs, shutting off most of the chip except when it is needed to perform a function may provide the greatest leverage.

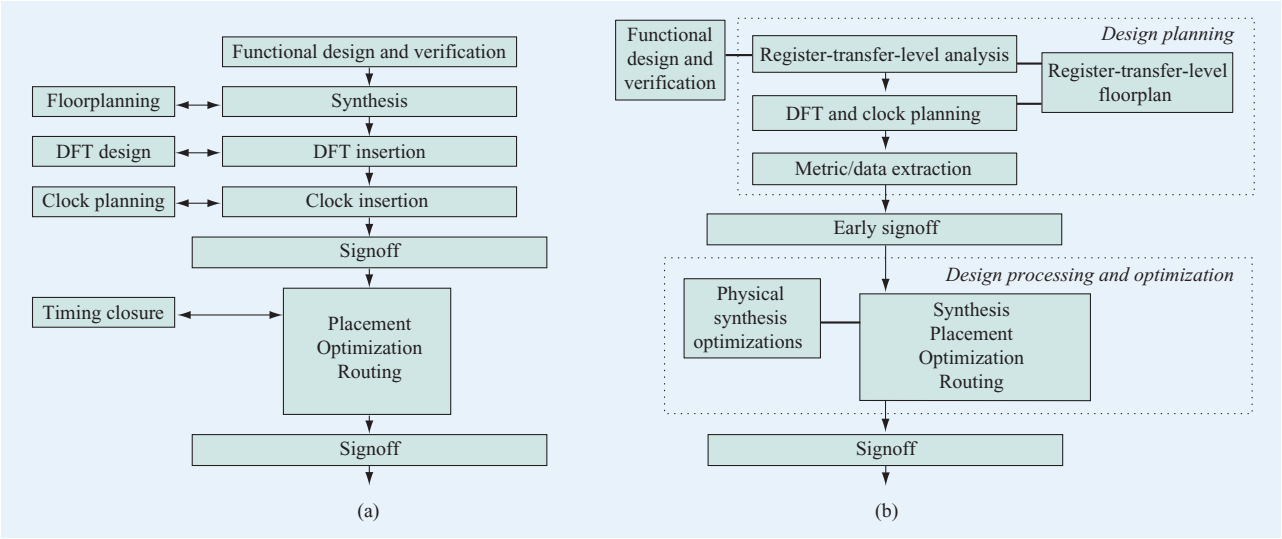

Figure 8

(a) Traditional ASIC flow; (b) improvement in design-planning flow.

As mentioned earlier, IBM customers already had reasons for partitioning large ASIC/SoC designs into a hierarchy. The need to design different parts of the chip to operate at different voltage supply levels could now be added.

Test efficiency and ac effects

With increasing raw chip capacity for a given test methodology comes the need for increased test data to contain the required test-pattern sets. With the effects of shrinking geometries of device and interconnect scaling, and the need for ever-increasing design performance, comes the need to exercise and detect time-dynamic or transitional defects, and to limit chip power consumption during the test application.

Chip designers and silicon vendors have traditionally approached the issue of test data reduction through methods such as logic BIST (LBIST) [31]. Although LBIST structures are highly effective in reducing the amount of data required for testing, the effectiveness of LBIST is reduced by logic structures resistant to random test patterns (high-fan-in structures, decoders, and comparators are examples). Whereas IBM ASIC chip test coverage levels nearly always exceed 99%, the issue of random resistance often limits LBIST coverage to the range of 85%–95% or less.

To meet the needs of reduced test data volume while maintaining product quality, IBM has devised enhanced versions of the traditional LBIST approaches. First, our ASIC chips have recently been provided with a built-in signature compression capability. Traditional IBM test patterns are input to the ASIC; however, all scan data pins (traditionally pairs of scan data pins support a single scan chain and scan-in and scan-out ports) are now devoted to scan-in ports, the number of scan chains is doubled, and scan-chain bit lengths are halved. The test results for the doubled set of scan chains are shifted into the signature register at the end of each test. This approach significantly reduces the data volume for test through the reduced size of the test results (using signature compression), and halved scan-chain lengths. Also, since the signature register is available as a parallel chip output, diagnostics are greatly simplified compared to traditional LBIST methods.

Shrinking device geometries increase the exposure of the chip to dynamic transition errors, or ac defects. IBM employs an ac test methodology for the highest-performance designs, whereby scan-based test patterns are deployed to expose specific nodes for transition (rise/fall) observability, and the tester applies pulses to the test clocks of programmable width and edge separation. The observed transition performance is compared for compliance within the delay range anticipated by the timing sign-off engine. Failures represent ac defects. IBM will deploy ac tests for increasing the number of ASIC chips in advanced technologies, with a focus on increased automation and reduced manual setups.

Conclusion

In developing ASIC methodologies, IBM has balanced the use of its own internal tools and infrastructure with key third-party vendor tools and industry-standard methods.

This enabled IBM to initially enter the market with a combination of first-time-right quality leadership, along with a front-end design environment familiar to most external ASIC designers.

Continued methodology improvements throughout the 1990s, along with strong process technology leadership, enabled IBM to achieve and sustain a leading market share in the OEM ASIC business. Area-array I/O capability gave IBM a clear lead in signal I/O accessibility to ASIC chips. Improvements in placement and routing tools, placement-based synthesis flows, key clocking improvements, and dynamic image generation substantially reduced design TAT. Pushbutton design-for-testability methods and tool infrastructure initiatives further addressed both TAT and usability issues.

Advancing the design methodology in the near future will embrace three basic principles:

1. Increased early, front-end focus in the design flow.
2. Expansion of technology and product offerings.
3. Extension of base silicon technology leadership.

The front-end focus provided by design planning and RTL signoff will move design issues traditionally addressed in the back end of the design flow to front-end design, where the leverage over performance, die size, power, and TAT is the greatest. New offerings will be enabled through the use of low-power design methods, reduced die size requirements and mixed terrains, and new system-on-a-chip capabilities that include analog and mixed-signal design. Base silicon technology leadership will be extended through a number of enhancements that address nanometer-level physical and electrical issues. Test enhancements will focus on containing the huge number of patterns required for increasing chip densities and addressing issues of ac defects and power consumption during test. The tool infrastructure is a fundamental enabler with respect to most of the above requirements, and the fully interoperable IBM tool infrastructure, as well as its established file interchange protocols with third-party tools, will realize these improvements in the IBM Blue Logic ASIC/SoC methodology flow.

Acknowledgments

The authors recognize the achievements of numerous teams whose contributions advanced IBM to a position of ASIC OEM leadership. In particular, the authors recognize the IBM Blue Logic ASIC Methodology, Technology Product Development, Embedded Product Development, and Product Engineering organizations, for developing a first-time-right design methodology; the IBM EDA organization and IBM Research and university partners, whose design capacity, algorithmic strengths, and technology correlation led to the development of a unique integrated design tool; the Research Institute of Discrete Mathematics of the University of Bonn, Germany, which developed timing-driven placement, optimization, and routing capabilities that have enabled design of some of our most complex ASICs; the IBM Design Centers, where methodology, tools, and design process management were applied to ASIC/SoC design; and the numerous IBM product development locations, including Rochester, Austin, Poughkeepsie, and Boeblingen, whose models of successful design execution have played a large part in the definition of methodology, tools, and design process used for ASIC design.

References

1. Cadence® "AMS Designer," © 2002 Cadence Design Systems, Inc.; see *http://www.cadence.com/products/amsdesigner.html*.
2. "Design Process Automation," © 2001 Interweave Tech Corporation; see *http://www.interweavetech.com/*.
3. Get2Chip®, "Get2Chip and Prolific Liquid Cell Study," © 2001 Get2Chip.com, Inc.; see *http://www.get2chip.com/docs/white_papers/g2c-lci.asp*.
4. eSilicon®, "eSilicon Access," © 2001 eSilicon Corporation; see *http://www.esilicon.com/services/access.html*.
5. Artisan Components®, "Process-Perfect Products," © 2001 Artisan Components, Inc.; see *http://www.artisan.com/products*.
6. *IEEE Standard 1076-1993*, "IEEE Standard VHDL Language Reference Manual," © 1998–1999 IEEE; see *http://standards.ieee.org/reading/ieee/std_public/description/dasc/1076-1993_desc.html*.
7. *IEEE Standard 1364-1995*, "IEEE Standard Hardware Description Language Based on the Verilog® Hardware Description Language," © 1998–1999 IEEE; see *http://standards.ieee.org/reading/ieee/std_public/description/dasc/1364-1995_desc.html*.
8. "Memory Arrays in IBM ASICs," IBM Application Note, © 2001 International Business Machines; see *http://w3asics.btv.ibm.com/*.
9. J. Czilli and A. Nordstrom, "Synthesizing a 650K Gate Deep Submicron ASIC," presented at the 1997 Synopsys Users Group (SNUG '97), San Jose; see *http://www.snug-universal.org*.
10. J. J. Engel, T. S. Guzowski, A. Hunt, D. E. Lackey, L. D. Pickup, R. A. Proctor, K. Reynolds, A. M. Rincon, and D. R. Stauffer, "Design Methodology for IBM ASIC Products," *IBM J. Res. & Dev.* **40,** No. 4, 387–406 (July 1996).
11. S. Oakland, J. Monzel, R. Bassett, and P. Gillis, "An ASIC Foundry View of Design for Test," *Proceedings of the IEEE International Test Conference*, 1994, Test Synthesis Seminar addendum, paper 4.2.
12. V. Chickermane, B. Koenemann, T. Guzowski, T. W. Williams, A. Sullivan, and S. Oakland, "DFT: Test Synthesis and Beyond," *Proceedings of the IEEE International Test Conference*, 1994, Test Synthesis Seminar addendum, paper 3.3.
13. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science* **220,** No. 4598 (1983).
14. J. Vygen, "Algorithms for Large-Scale Flat Placement," *Proceedings of the 34th Design Automation Conference*, 1997, pp. 746–751.

15. G. L. Smith, R. J. Bahnsen, and H. Halliwell, "Boolean Comparison of Hardware and Flowcharts," *IBM J. Res. & Dev.* **26,** No. 1, 106–116 (January 1982).
16. G. E. Moore, "Cramming More Components onto Integrated Circuits," *Electronics* **38,** No. 8, 114–117 (1965).
17. H. B. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley Publishing Co., Reading, MA, 1990.
18. D. Lackey, "Applying Placement-Based Synthesis for On-Time System-on-a-Chip Design," *IEEE Custom Integrated Circuits Conference*, 2000, pp. 121–124.
19. Cadence® "Physically Knowledgeable Synthesis," © 2002 Cadence Design Systems, Inc.; see *http://www.cadence.com/products/pks.html*.
20. Synopsys® "Unified Synthesis and Placement," © 2002 Synopsys, Inc.; see *http://www.synopsys.com/products/unified_synthesis/unified_synthesis.html*.
21. J. Darringer, E. Davidson, D. J. Hathaway, B. Koenemann, M. Lavin, J. K. Morrell, K. Rahmat, W. Roesner, E. Schanzenbach, G. Tellez, and L. Trevillyan, "EDA in IBM, Past, Present and Future," *IEEE Trans. Computer Aided Design* **19,** No. 12, 1476–1497 (December 2000).
22. A. Hetzel, "A Sequential Detailed Router for Huge Grid Graphs. Design, Automation, and Test in Europe," *Proceedings of the IEEE International Conference on Computer Aided Design*, 1996, pp. 332–338.
23. C. Albrecht, B. Korte, J. Schietke, and J. Vygen, "Cycle Time and Slack Optimization for VLSI Chips," *Proceedings of the IEEE International Conference on Computer Aided Design*, 1999, pp. 232–238.
24. M. Kuzawinski, "High-Density Package Applications for Wire Bond and Flip Chip," *Proceedings of the Semiconductor Packaging Technologies Symposium—SEMICON West '99*, 1999.
25. *IEEE Standard 1149.1-1990*, "IEEE Standard Test Access Port and Boundary-Scan Architecture," © 1998–1999 IEEE; see *http://standards.ieee.org/reading/ieee/std_public/description/testtech/1149.1-1990_desc.html*.
26. V. Chickermane, D. Lackey, D. Litten, and L. Smudde, "Automated Chip-Level I/O and Test Insertion Using IBM Design-for-Test Synthesis," *IBM Micronews* **6,** No. 2, 18–22 (Second Quarter 2000).
27. D. Lackey and J. Morrell, "Interoperability, an IBM Perspective," IBM Microelectronics Division 2001, presentation at the 2001 Design Automation Conference Interoperability Workshop.
28. *IEEE Standard 1481-1999*, "Integrated Circuit (IC) Delay and Power Calculation System," © 1999 IEEE; see *http://standards.ieee.org/reading/ieee/std/dasc/1481-1999.pdf.*
29. D. Lackey, "Design Planning Methodology for Rapid Chip Deployment," *Proceedings of the Eighth IEEE/DATC Electronics Design Processes Workshop (EDP 2001)*, pp. 111–116.
30. A. Dean, D. Garrett, M. Stan, and S. Ventrone, "Low Power Design for ASIC Cores," *VLSI Design,* pp. 1–15 (2000).
31. P. Bardell and W. McAnney, "Self-Testing of Multiple Chip Modules," *Proceedings of the IEEE International Test Conference*, 1982, pp. 200–204.

Received January 24, 2002; accepted for publication August 15, 2002

George W. Doerre *IBM Microelectronics Division, East Fishkill facility, Hopewell Junction, New York 12533 (doerrg@us.ibm.com).* Mr. Doerre manages ASIC and EDA strategy for IBM Microelectronics Product Development. In 1980, he joined IBM in Burlington, Vermont, working on advanced DRAM development. Since then, he has held management positions in VLSI product and process development and ASIC design methodology development and integration. He holds M.S.E.E./C.S. and M.S. physics degrees from MIT.

David E. Lackey *IBM Microelectronics Division, Burlington facility, Essex Junction, Vermont 05452 (delacke@us.ibm.com).* Mr. Lackey is a Senior Technical Staff Member in the IBM ASIC Product Development group, responsible for ASIC methodology development. In 1978 he joined IBM in Poughkeepsie, New York, in the Mid-Hudson Valley Development Laboratory. Since 1994, Mr. Lackey has developed leading-edge design methodologies for IBM and external ASIC designs. He received a B.S.E.E. degree from Rensselaer Polytechnic Institute in 1978 and an M.S.C.E. degree from Syracuse University in 1983. He is a member of IEEE and Eta Kappa Nu.