RC 21829 (98249) 11 September 2000 **ComputerScience**

IBM Research Report

EDA In IBM: Past, Present and Future

John Darringer, Evan Davidson, David Hathaway, Bernd Koenemann, Mark Lavin, Joseph Morrell, Khalid Rahmat, Wolfgang Roesner, Erich Schanzenbach, Gustavo Tellez, Louise Trevillyan.

IBM T. J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - T. J. Watson - Tokyo - Zurich

EDA in IBM: Past, Present and Future

Authors: John Darringer, Evan Davidson, David Hathaway, Bernd Koenemann, Mark Lavin, Joseph Morrell, Khalid Rahmat, Wolfgang Roesner, Erich Schanzenbach, Gustavo Tellez, Louise Trevillyan.

Abstract

Throughout its history, from the early 4-circuit gate-array chips of the late 60's to today's billion-transistor multi-chip module, IBM has invested in tools to support its leading-edge technology and high-performance product development. The combination of demanding designs and close cooperation among product, technology and tool development has given rise to many innovations in the electronic design automation (EDA) area and provided IBM with a significant competitive advantage. This paper highlights IBM's contributions over the last four decades and presents a view of the future, where the best methods of multi-million gate ASIC and gigahertz microprocessor design are converged to enable highly productive system-on-a-chip designs that include widely diverse hardware and software components.

I - Introduction

Advances in design automation usually arise during periods of extreme stress created by a product group designing a high-performance product, using the latest technology with an aggressive schedule. If at the same time there is a strong collaboration among the product, technology and tool developers, together with a willingness to take some risk, sparks can fly. IBM has long been an incubator for innovation in EDA in their Research and Development organizations and the sections that follow describe four particularly productive environments. First was the development of the bipolar mainframe machines of the 80's. This period in IBM produced a remarkable foundation for predictable and efficient design of complex systems using regular chip and package layouts with a highly-automated design system. In the 90's, processor design evolved into today's competitive battle to produce extremely complex "microprocessor" systems, while exploiting custom layout and new circuit families to operate at gigahertz clock frequencies. This shift in design style has led to the development of new classes of transistor-level analysis and optimization tools in IBM along with a much more flexible and extensible design system, which allows more rapid reaction to ideas of gifted designers. The 90's also gave rise to IBM's development and commercial offering of extraordinary high-performance and high-density application-specific integrated circuits (ASICs). The combination of leading-edge technology and tremendous time-to-market demands have created a highly efficient design methodology, supported by a tightly-integrated set of modular tools operating incrementally on a shared, in-memory data model and capable of supporting 40 M-gate chips. The future promises to be even more exciting. Technology advances will provide larger chips with a wide diversity of components that can be integrated into a single chip or multi-chip package. Success in this future will require a highly integrated design system capable of allowing a designer to optimize at many levels of abstraction from hardware and software behaviors to devices and shapes, all within an even shorter schedule.

II - EDA Milestones in IBM

IBM has a rich history of contribution to the field of design automation that spans four decades. Tables 1, 2 and 3 list the more significant advances in verification, design and test, respectively. The following sections provide more insight into the nature of these advances.

A - Verification

Progress in verification has always been driven by the largest and most complex products. During the 80's these designs were the S/370 class mainframes. The synchronous design style used for the 3081 project [1] laid the foundation for a verification methodology for the next two decades. The 3081 was the first product to rely on simulation for functional verification and found 84% of all logic design problems before the hardware was developed (today's standard is above 96% [2]). The 3081 verification methodology was based on the following objectives:

- Design specification in a high-level (RT-Level) hardware design language
- Complete separation of functional verification from timing verification
- Cycle-based simulation
- Simulation coverage analysis using properties of the high-level language specification
- Formal Boolean equivalence proof of gate-level implementation vs. high-level design

The S/370 design team realized the benefits of these innovations long before the rest of the industry:

- A formal functional specification at the outset of logic design.
- Compact, high-performance simulation models
- Coverage measures for clear feedback on simulation quality and verification completeness
- RT-level verification wit proven equivalence to the gate-level design avoided costly gate-level simulation

Throughout the 90's, microprocessors and the associated systems were the drivers of the verification technology. The function to be verified in these projects was at least as complex as the driver designs in the 80's. Key factors for the evolution of the verification methodology include:

- Explosion of processor and system complexity
  - Large scale SMP systems
  - Superscalar processors with out-of-order and speculative execution
  - Custom circuit implementations
- Proliferation of Unix (AIX) workstation
- Emergence of Industry-Standard VHDL and Verilog
- Development of practical functional formal verification

Design Specification

IBM developed a series of hardware-description languages [3] that focused on abstract specification of control flow, deliberately neglecting any correlation with the physical structure of the hardware. As logic synthesis was applied to hardware description language (HDL) designs, the style of HDL entry began to change from purely functional coding for simulation to a style that would let the designer "steer" synthesis by providing structural information. IBM was among the first supporters of the DoD effort to create VHDL, even though its focus on an event-driven simulation paradigm did not fit well with IBM's established synchronous design methodology. For several years the internal languages provided superior capabilities, but the desire for an open standard HDL led to improved support and widespread use of VHDL. Today VHDL and Verilog have fully displaced the earlier in-house languages.

Cycle-Based Simulation

Hardware simulation was traditionally done with simulators that used event-driven algorithms. IBM's synchronous design style allowed the separation of timing verification from functional verification. This enabled simulation that can evaluate the state of the simulated logic only by the end of each machine cycle. Such a zero-delay evaluation of the Boolean logic gates between state elements was organized in a rank-ordered fashion such that much of the overhead of the event-driven algorithm was eliminated. This cycle-based simulation method increased simulation speed by one to two orders of magnitude beyond event-driven simulators, with the additional advantage that performance and memory requirements scaled at most linearly with the problem size. Cycle-based simulation evolved from early forms of direct interpretations of the hardware description, to an algorithm that used rank-ordered logic evaluation [4] and on to a simulator that applied limited compiler techniques to map many Boolean operations into single machine instructions [5],[6]. Most of the complexity of these cycle-simulators was in the process that produced executable machine code for the model.

The development of the RS/6000 workstations provided an ideal target for compiled-model simulation. A compiled cycle-simulation model consists mostly of Boolean operations that can be executed in a single cycle. The new Unix(AIX) environment also prompted a more efficient implementation of all the algorithms for model creation. The new simulator was named "Texsim" [7], [8]. Texsim gained its efficiency from a Boolean network database optimized for mapping HDL descriptions to machine instructions. Model build is related to logic synthesis and the code-generation back-end of a programming language compiler but has different tradeoffs than these. Turnaround time is important as large optimized simulation models of 100-200MB in size need to be generated in minutes.

The second generation of Texsim, implemented in the mid-90's, included two innovations: 1) a multi-value (0,1,x,z) code-generation mode was developed which largely eliminated the need for any event-driven logic simulation and 2) the idea of using the vectored, 32-bit word for the evaluation of 32 'parallel patterns' was developed in a new way [9]. Model size had exploded not only because of denser chip technology, but also because the emergence of microprocessor based SMP systems. The 'parallel instance' feature allowed the automated vectorized packing of multiple instances of the same module in a model by 1997. For example, while a 601 PowerPC (1 million transistors) model takes less than 2 minutes to build from HDL source and shows a simulation throughput of 350 cycles/sec, it is possible to build an 8-way Power3 (16M transistors per processor, or 128M transistors per system) model in less 15 minutes with a resulting throughput of 10 cycles/sec (all numbers based on an RS/6000 595). All of IBM's microprocessor systems have been simulated with Texsim since 1992 [8].

The availability of inexpensive workstations resulted in a workstation under every designer's desk which in turn led to the concept of a 'simulation farm'. Simulation jobs are submitted through a batch system The advantage was to be able to run testcases on hundreds and thousands of workstations in ideal parallelization and with direct scalability [7]. Even with today's large complex SMP models it is still possible to achieve 1 Billion simulation cycles over a weekend with software simulation alone.

Simulation Acceleration Hardware

An important component of the massive simulation horse power needed to verify IBM's complex systems were special purpose hardware accelerators. After developing two early models [10], [11], a robust production system was developed, the Engineering Verification Engine (EVE) [12]. EVE used a massive network of Boolean function processors which each were loaded with up to 8192 logic instructions. Typically, each run through the sequence of all instructions in all logic processors in parallel constituted one machine cycle, this implementing the cycle-based simulation paradigm.

The theoretical speed of EVE was many orders of magnitude faster than any software implementation - 2.2 billion gate evaluations per second. In practice, throughput in cycles/sec (cps) for any given processor model determined the value of EVE for a project. Throughput was determined by the slowdown of the engine by model load, setup, results analysis, and most importantly by the amount of interaction between engine and compute host. The importance of spending most of the runtime in the engine at full speed led to innovations like synthesis of checker and testcase driver logic into 'virtual hardware' [12]. A multiprocessor model with the full storage hierarchy and I/O boards achieved between 250cps to 1000cps compared with 0.5cps for the software model run on an S/370 mainframe [2]. At the peak use there were nine EVE machines shared among IBM's product designs [5], [8], [13].

In the late 90's, "Awan", was built as a low-cost system which improved on both the capacity and performance of EVE. Awan is much like the EVE machine, but it is made with smaller, faster components and has a much-improved interconnection strategy. Models exceeding 31 million gates have been simulated. Speed depends on the configuration, model size, model complexity, and the amount of host interaction. The raw model performance of the Power4 chip running on Awan exceeds 2500 cycles/sec. Awan is marketed by Quickturn under the name Radium.

Utilizing the base EVE concepts, a hyper-acceleration and emulation machine called ET3 [14] was developed in IBM's CMOS technology. ET3 uses logic processors which evaluate 3-way input gates. In contrast to AWAN, ET3 has a larger number of processors and a lower depth of sequential 3-way-gate instructions per processor (256 vs. 8k in EVE or 128k in AWAN). The resulting higher degree of parallelization leads to dramatically higher speed (50k-1M cycles/sec), but at a much higher hardware price. The model build for the accelerator and emulator system taps into the Texsim system using it as a common front-end. This makes the selection of the target simulation engine a simple option for the user. Acceleration has its traditional place in IBM's verification flow. Emulation has been successfully used in graphics processor and MPEG projects. The breakthrough of this technology in the microprocessor and server system space occurred after the latest capacity improvements [14]. ET3 is marketed by Quickturn under the name CoBalt.

Test Program Generation

While innovative simulators and accelerators can provide enormous simulation capacity, it is just important to use these cycles wisely through intelligent test case generation. For small scale simulation at the module or unit level, individual testcase drivers with random stimuli were used. Since the design being simulated was usually a processor it was natural to use processor instructions, loaded into the memory of the simulated machine, as the testcases. Early on, these test programs were either manually developed or derived from code fragments from previous machines. Random test case generation was also used explore subtle errors [15] and was refined to produce test targeted at specific conditions.

A lightweight System Assurance Kernel (SAK) [5] was used in bring-up labs to test the new systems. SAK allowed the development of a diverse set of testcase drivers or generators which would dynamically generate test instruction streams. The machine would execute the test program and SAK would check the machine state vs. the predefined correct result. With the emergence of the EVE hardware accelerator technology it became viable to run parts of the lab bring-up process in cycle-simulation before the design was committed to silicon. Several projects used this process for full-system simulation [16].

A key focus for test program generators in the 90's is the increased complexity of the micro-architectures which employ ever more advanced schemes to improve processor performance, such as deeper pipelining, branch predictions, and more aggressive superscalar and speculative execution. To address these advances in design complexity with higher quality tests, 3 different generators were developed. The combination successfully advanced high quality testing into the realm of complex SMP systems and allowed the efficient utilization of the software simulation workstation farms.

AVPGEN was specifically developed for S/390 verification [8], [17]. It uses symbolic instruction graphs as a format for the verification engineer to specify templates for test programs to be generated. The templates are a powerful format to target classes of test cases. Symbolic values are used to express constraints and value dependencies and are also used to leave the generator options to let its constraint solving algorithm choose concrete values as late as possible to reach interesting corner cases.

Genesys [18] separates the generator into several distinct components. An independent reference model provides an instruction-set execution model of the machine An architectural model both encapsulates and abstracts machine architecture specifics in the form of instruction trees. Testing knowledge is encapsulated in C routines written by verification engineers and called by the generators at appropriate times during the traversal of the architectural instruction trees. Tree traversal is at the heart of the generation process. Constraint solvers are guaranteed to generate correct values as they are bound to the instruction tree. Genesys has been used successfully on the AS/400 and RS/6000 processors. [18] quantifies the beneficial effect of creating higher-quality test cases.

MPTG [19], [20] is another generator that addresses multiprocessor cache coherency verification. The reference machine model of MPTG is a combination of memory hierarchy and its associated coherency protocols and is declarative. The test specifications control the occurrence of specific sequences of cache events. In contrast to Genesys, test results are verified by inspecting storage locations in the memory hierarchy and monitoring coherency. Since its creation, MPTG has been used in all PowerPC and PowerPC-AS system verification projects.

Boolean Equivalence Checking

IBM has explored formal methods since the 70's [21] and succeeded in applying them to a product design in 1978 [22]. But the tool that had the most impact in IBM was the Boolean equivalence checker called SAS for Static Analysis System [23]. It was based on the use of Shannon's expansion. It had a simple but powerful user-interface which allowed the designer explicit control of equivalence-point, or cut-point, selection and other bookkeeping measures to address the inevitable problems with large designs. SAS was remarkable for its very early use of formal methods in production computer design. For large synchronous designs, such as the 3081 and ES/9000, SAS eliminated the need to do functional verification at the gate level.

With the emergence custom transistor implementations, a new approach was necessary to guarantee the correctness closure on which the complete HDL-level methodology is founded. One of the contributions of Verity [24] was to use a mixed-mode circuit extractor that is adaptable to a wide variety of circuit design styles. Extraction was

combined with the application of consistency checks which validate the extraction model. The extracted model was verified against an HDL specification using a variety of algorithms which combine the application of BDDs with graph hashing, automatic insertion of multiple cut points and a controlled elimination of false negative results caused by the cuts. A unique innovation of Verity was that it uses different algorithms seamlessly to prove equivalency. These different algorithms, implemented as separate engines, play out different tradeoffs to the comparison problem depending on how structurally different the two input designs are.

While Verity is used on flat netlists, its capability to support hierarchical formal verification ties well into the design flow. Leaf cells of the design hierarchy are completely compared, and these results are used in processing the higher levels of the design hierarchy. Since custom-circuit implementations often exploit macro input constraints, Verity supports an assume-guarantee scheme for these constraints: on macro-inputs the constraints are assumed in the equivalency check, on the macro-output they are required to hold and are therefore proven by Verity.

Model Checking

Extending the reach of formal methods became practical with the success of BDD-based model checkers. Based on the Symbolic Model Verifier, SMV [25], 'RuleBase' [26] was developed in the early 90's. Many RuleBase innovations involve methods that address the model size problem of BDD-based model checking. As a result, RuleBase was successfully applied to designs [27] like bus bridges, cache controllers, bus interface units and functional units of microprocessors, and pushed this new technology into the mainstream verification process.

Micro-Architectural Modeling

Today's HDL models are aimed at describing implementations and fail at capturing a designer's real intent. It is necessary to raise the level of abstraction and create a model that captures the design at the micro-architectural level. Many verification tasks could be improved with such a 'high-level' model: simulation (speed), formal verification (model size, easy separation of control logic), coverage (obvious structures to instrument for coverage models), and test program generation (project specific reference model for focused test generation). This high-level, executable specification would have benefits for the overall design process beyond verification. At IBM, work is proceeding based on early success with a modeling framework, called "Faust". This C/C++ environment enabled the micro-architects of the Power4 project [28] to write an efficient, concise micro-architectural model. The system allowed the designers to use VHDL for the structural specification of the upper levels of the design hierarchy. Sharing the same source between physical and high-level functional design is highly desirable. The lower-level of a Faust model is C/C++ code, which relies on the support of base class library. The library supports model partitioning, simulation control flow, built-in elements like latches and performance-related constructs.

While development of the approach continues, the initial experience of micro-architectural modeling on the Power4 project was very successful. Not only were reliable performance measurements derived from the model, but the verification process of Power4 benefited in major way:

- A machine-readable, executable specification proved early on that the processor over all 'hangs together'.
- Verification infrastructure and the verification team got an early start with an executable model that was available 1 year earlier than the actual HDL model.
- The exercise of developing the model increased the team's understanding of the design, leading faster to the more robust RTL implementation.

	Table 1 - Verification Milestones in IBM	
1978	First formal micro-code verification of product	MCS[22]
1978	Symbolic execution used for software and hardware verification	Effigy[21]
1982	First production use of cycle-based simulation	EFS[1]
1982	First simulation accelerator	YSE[11], LSM[10]
1982	Production use of automatic Boolean Equivalence Checking	SAS[23]
1982	Production use of function test program generation in simulation	SAK[16]
1990	Production use of cycle-base simulation on RISC workstation farms	TEXSIM[13]
1990	Production use of biased-random test program generation	RTPG[18]
1994	Boolean equivalence checking extended to transistor level	Verity[24]

1994	Production use of Model checking	RuleBase[26]
1995	Dynamic test program generation	Genesys[18]
1996	Multi-value and vectored SMP cycle-based simulation	TEXSIM[8]
1999	Ultra-parallel high-performance emulation machine	ET3[14]

B - Logic Design

Timing Analysis

In the early 1970's simulation provided the major means of timing verification, and it was clear that a new capability was needed to avoid reliance on patterns and exponential blowup. To avoid these problems, the PERT-based Timing Analysis (TA) program was developed [29] as part of the IBM 3081 design verification methodology. In order to give significant run-time improvements over path enumeration, TA used a *block-oriented* algorithm in which the blocks in the design were topologically sorted to allow a single-pass computation of all signal arrival times and required arrival times, their differences giving a *slack* value on a node to indicate timing criticality. Static timing analysis techniques could be applied so successfully because of the design discipline imposed by the LSSD test methodology, which clearly separated clock and data signals and enforced a strictly clocked synchronous design. TA propagated rising and falling timing values separately to model asymmetric circuit characteristics. A crude form of statistical timing analysis was provided in which the mean, sigma, and sigma squared of the arrival times were all propagated and combined with correlation information to compute slacks and perform tests at storage elements. As wire delay became more significant a change was made from considering only block delays to considering both block and net delays, with a set of timing values computed on each block port rather than on each net. To accommodate hierarchical timing analysis, abstraction capabilities were also introduced.

TA expanded the blocks in the design into a set of interconnected delay blocks, similar to the expansions used for test generation., Each block in the expansion had a single delay computed by an equation specified in the delay rule. The set of available delay equations and their corresponding delay coefficients were fixed, but tended to be extended over time as new delay dependencies arose. These delay coefficients were computed from curve fits to circuit simulation results. Delays were calculated for the 3 sigma worst case process and the 3 sigma best case process to ensure that all functional chips could be used in machines, since no delay sorting of chips was done. The SRAMs and embedded logic macros (e.g., register stacks) were handled with behavioral rules that were manually coded to provide the required functional and timing information. Paths that left the chip had partial delays calculated. These delays would subsequently be used by the TA program when all of the package interconnection data were supplied.

Meanwhile, in the CMOS domain, a separate timer had been developed to support the more complex clocking schemes used in CMOS designs [30]. In 1990, motivated by demands for more accurate timing, consistent timing throughout the design process, convergence of the timing analysis approaches, and workstation-based tools, IBM embarked on the EinsTimer [31] system. EinsTimer was developed as a timing utility rather than as a standalone tool. As such, it could be used standalone or as part of a variety of tools, including logic editing, logic synthesis, and placement. An incremental capability was provided which automatically invalidated timings when design changes which affected timing were made, and which minimized the recomputation needed when new timing information was required [32]. This efficient incremental capability enabled closer integration of synthesis, timing and physical design. To better support transparent latch design, EinsTimer was able to break loops which violated the acyclic graph assumption on which block-oriented static timing analysis depended, introducing new constraints to safely bound the timing at these loop-breaking points. It could then perform an iterative slack stealing across the loop breaking points to further reduce timing pessimism [33]. In EinsTimer early and late timing values were propagated separately using different delays reflecting the expect delay variation within a chip, to avoid the optimistic assumption of perfect on-chip delay correlation. This could introduce unnecessary pessimism when a common clock path fed the launch and capture latches of a critical path. As clock delays became more significant this pessimism became unacceptable, and capabilities were added to selectively remove this pessimism when needed [34].

To handle the rapidly growing set of delay dependencies and to isolate them from the underlying timing analyzer, the Delay Calculation Language (DCL) was developed. DCL provided this flexibility through a mechanism whereby the delay rule could make queries back to the timing analyzer for necessary values upon which delay values depend.

The DCL language has been accepted as an IEEE standard 1481 and has been expanded to include power calculation. More information is available from the Silicon Integration Initiative at www.Si2.org.

Early Synthesis

IBM has a long history of contributions to logic synthesis. Beginning in 1953 with Karnaugh Maps [35], through Alert[36], MINI [37] and YLE[38], IBM made progress in easing the task of designing logic. While these methods provided needed improvements, they were all PLA-based and therefore suffered from exponential behavior. They also did not match the dominant, library-based design style currently in use.

Production Synthesis

In 1979, development of the Logic Synthesis System (LSS) [39] was begun. A key observation was that the design did not need to be optimal - after all, the manually designed logic wasn't - but it did need to meet the same requirements (e.g. of speed, area, testability) that the human designers had to meet. Rather than basing the system on PLA minimization, the team proposed to use *local transformations* to simplify the logic. This would avoid an exponential run time, fit well with the multilevel library-based design style, and would avoid the complete structural collapsing associated with forming PLAs. LSS optimized logic at an abstract Boolean level and followed this by technology-mapping and timing correction scenarios to convert the design into the technology library primitives and to achieve timing constraints.

LSS was first used in production in 1982 on the bipolar chips used for the ES/9000 mainframe. In production use, it quickly became evident that it was necessary to have an incremental logic timer integrated with the logic synthesis system to allow it to make area-timing tradeoffs. Timing correction was applied at the early Boolean level to restructure the logic and also after technology mapping to take advantage of technology features. It was also realized that there were some problems, such as redundancy removal, which were global phenomena and would require solutions not limited to local transformations. This lead to pioneering work on redundancy removal [40], and on global flow analysis [41], which contributed to rewiring methods used in logic synthesis today.

Second Generation Synthesis

In 1989 it was decided to implement a new, workstation-based logic synthesis system An important feature of this new system, BooleDozer [31], was that its internal data model was also used within the IBM timing analysis and physical design systems. The data model provided general object annotation capabilities and a callback mechanism to notify applications of model changes. This eased the integration of multiple incremental applications operating on the model, and positioned BooleDozer for the integration of logical and physical design.

In the realm of technology-independent optimizations, BooleDozer provided improvements to the redundancy removal process were made by integrating a full-feature test generation program [42] within synthesis. For technology mapping, the limited pattern generation and covering algorithms used in LSS were extended and refined. Pattern generation was more aggressive, and the covering algorithm used a tiling method to choose the final implementation. Both programs were sensitive to timing as well as to area.

Timing correction was a particular emphasis in BooleDozer. An important advance was the development of a method to improve its decisions about where correction transformations could most profitably be accomplished. Other improvements in gaining timing closure were also incorporated into BooleDozer. Some examples are the use of recovering for timing, in which a timing-critical section of logic is translated back to technology-independent form and then re-technology mapped for better timing, and the use of checkpoints and hill climbing.

Incremental synthesis was another important feature in BooleDozer. One downside of using automated synthesis was its tendency to be unstable in the face of "small" design changes. A designer would make what seemed to be a trivial change, and the synthesized results might be completely different. Incremental synthesis overcame this difficulty. by reading both the previous and changed designs and "protecting" unchanged logic, encouraging minimal changes to the design. This was especially useful when there had been considerable downstream work in tuning the design, and was also helpful in easing the task of verification.

In the mid 90's, the emphasis on very high-performance circuits and the advent of sub-micron technologies caused some fundamental changes in logic synthesis. While it had always been a challenge to synthesis to obey timing constraints, the new, more stringent timing requirements and the shift of delay from the gates to the wires called for new synthesis techniques. The BooleDozer team responded with work in *synthesis of dynamic logic* [43], to allow synthesis to exploit the advantages of specific circuit families, such as domino logic; *gain-based synthesis* to improve the timing characteristics of the design, especially by fanout correction [44] and wavefront technology mapping [45], and to allow simplified libraries; and *transistor-level synthesis*, to optimize time and area on critical segments at the detailed transistor level.

Placement-Driven Synthesis (PDS)

PDS was an important new technique that combined, BooleDozer, the EinsTimer timer, and physical design capabilities to overcome the problems of achieving timing closure To merge logic synthesis and placement, it was necessary to have both operate incrementally and independently. The process started as a pure synthesis process, but the goal was to bring placement in as soon as possible. It is especially important to have physical information during significant logic restructuring phases in order to control wire lengths and delay. An example of restructuring enabled by PDS is physically-based buffer insertion capabilities [46]. Standard buffering methods considered only logical connectivity, but the new capability was based on Elmore delays, dynamic programming, and a sink to source walk of the global route for the net to be buffered.

The overall strategy of PDS in merging the two applications was to place the logic on a grid. Initially, the grid regions were large, so the granularity of the placement was very coarse. As the process continued and the logic began taking on its final form, more cuts were done to reduce the granularity, and increase the accuracy, of the placement. Wire lengths [47] were estimated using Steiner trees.

To reduce the time required to achieve timing convergence on large chips, Parallel Hierarchical Timing Correction (PHTC) capabilities were developed [31]. An hierarchical design was read into BooleDozer processes running on several different machines, and a complete chip timing analysis was performed in each. Each process then selected a macro to work on (with locking to prevent selection collisions) based on the worst slack in the macro and on the number of times it has been chosen. The process improved the timing of the macro, treating the surrounding macros as frozen. When finished it wrote back the updated macro, read in any updated macros, and repeated the process. The incremental timing analysis capability in EinsTimer ensured that timing results were updated when new versions of macros were read in.

Behavioral Synthesis

The success of logic synthesis in raising designer productivity naturally led to the goal of raising the level of abstraction even more, from the register-transfer level to the behavioral level. Work on high-level synthesis began in 1984 and resulted in the HIS system [48]. Used throughout IBM, HIS provided a single port of entry for VHDL designs, which encouraged IBM designers to use higher levels of abstraction in their specifications. Unlike other dataflow-centered approaches, HIS emphasized the synthesis of efficient control structures. A major technical contribution was the work done in resource sharing [49], which used interleaved register and functional unit merging in a global clique-partitioning-based framework, accurate estimations of the costs of interconnect and unit merging, use of relative control cost and efficient false loop elimination. The results obtained showed significant improvements in the delay of designs, while also minimizing area.

C - Transistor-level Design

The earliest form of EDA software was developed to analyze and characterize high-speed computer circuits. In the early sixties, logic circuits were analyzed using ad-hoc equations. As circuits grew in size and complexity IBM pioneered the systematic use of EDA tools and revolutionized the way that circuit design was performed. The development of EDA in the circuit area started with computer programs and methods for network analysis, progressed to optimization of circuits using computers, and finally to complete automatic layout of circuits.

Over the last two decades the needs of IBM circuit and system designers have driven the development of circuit and transistor level tools. In the 80's, IBM engineers were designing chips in bipolar current-switch-emiter-follower

circuits in a gate-array structure with 4 levels of metal. Hence, the focus was primarily on accurate circuit simulation for relatively small circuits and accurate modeling of interconnect due to the reliance on mult-chip modules for integration. As designers began using CMOS VLSI with greater integration on chip, initially the approach adopted was to use standard cells with limited need for transistor level tools. In the late 90's, the demand for higher performance, with gigahertz clock speeds, both in the S390 class servers as well as PowerPC based UNIX servers, has led to a greater emphasis on custom design requiring the range of IBM tools to expand to include a myriad of transistor level tools, while still providing increased capabilities in traditional circuit and interconnect analysis.

Circuit Simulation

Franklin Branin at IBM Kingston was one of the first to point out how EDA was changing the modus operendi of circuit design [50]. He described the topology of a circuit as a linear graph, and superimposed an algebraic structure on the graph based on the interrelationships between nodes, branches and meshes of the graph. The algebraic structure could be compactly reduced to a matrix of equations, which was amenable to computer manipulation. This work represented the foundation of circuit simulation, which continues to be used today, with two decades of improvements to formulations and matrix solution techniques.

The "Sparse Tableau Approach to Network Analysis and Design" [51] was one such advance, and although this technique is not used in present day circuit simulators, this was the first complete incorporation of sparse matrix techniques into automated network optimization. Another major contribution was the Modified Nodal Approach (MNA) [52] which was a generalized formulation that enabled circuit simulators to handle current-dependent elements while improving program speed and memory utilization. MNA continues to be used today in most circuit simulators, including all present day SPICE simulators. With these advances came the first circuit simulator, ASTAP [53], which was widely used throughout IBM. With circuit sizes growing exponentially, there was an increasing need to continually improve speed and reduce memory requirements for circuit simulators. One breakthrough was development of "waveform relaxation" techniques which enabled partitioning of large circuits into smaller sub circuits, and thereby allowed independent analysis of these sub circuits [54].

In the 90's, the continuing focus on circuit simulation of ever larger circuits required the use of new numerical techniques as well as the leveraging of multiprocessing capability. Both these trends led to the incorporation of the waveform relaxation algorithm [54] in PowerSPICE, the simulator which is currently in use in IBM [55]. As the computer industry moved from bipolar to CMOS for high performance digital designs, and the area of conventional circuit simulation continued to evolve, a more approximate "timing simulation" technique which bridged the gap between logic simulation and detailed circuit simulation was started at AT&T Bell Laboratories with the development of MOTIS [56] for MOS devices. MOTIS was the first to incorporate table models to represent MOS devices, rather than simple '1' and '0' used in logic simulation, or the detailed, time-consuming evaluations of analytic equations used in analog circuit simulation. Timing simulation was introduced in IBM with SPECS [57], which used piecewise-constant device models and event driven simulation to provide speed and variable accuracy. Timing simulation was further advanced with the introduction of ACES [58], which incorporates piecewise-linear device models and a novel integration algorithm to improve performance and accuracy.

Interconnect Modeling and Signal Integrity Analysis

While breakthroughs were being made in the circuit simulation arena in IBM, a parallel effort was underway for accurate modeling of interconnect - the results of which were used for more accurate circuit simulation. IBM was a pioneer in the analysis of coupling noise, delta-I noise and timing delay which occur due to interconnect parasitics. Static capacitance and inductance computations which result in networks, which could be simulated practically, were first computed using the PEEC (partial element equivalent circuit) method [59], [60], [61], [62]. These were used for on-chip and off-chip delay or coupling or delta-I noise calculations. For packages, the electrical length of interconnect required a more complex transmission line analysis to compute delay and noise. IBM was one of the first companies to model package parasitics using both lossless and lossy transmission line analysis [63], [64]. The result of all these innovations was a CAD tool package COSMIC [65] which included tools for two and three dimensional capacitance, inductance and lossy transmission line coefficient calculations. COSMIC continues to be widely used within IBM for parasitic computation.

Until recently, noise analysis was performed only on small subsections of a macro using circuit simulators, as timing analysis used to be years ago. Harmony [66], the first exhaustive "static" noise analysis approach was initiated at IBM in 1996 and continues to be developed today. This allows large functional units, with tens of thousands of gates to be analyzed for all types of noise-related problems including coupling, charge-sharing and leakage. The analysis at the functional unit or macro level is encapsulated in noise abstractions that are used at the chip level, where coupling noise is calculated for all global wires, using fast model order reduction methods, and can be compared to the acceptable level of noise at any macro input to verify its susceptibility. In conjunction with the noise analysis tools, fast and highly accurate parasitic extraction tools have been developed by IBM both at the transistor level and the chip level. At the chip level IBM's extraction tool 3DX was one of the first tools to accurately extract coupling capacitances for noise analysis and frequency dependent self inductances for timing analysis [67].

Power Management and Distribution

At the chip level power supply collapse due to simultaneous switching and voltage drops is a major concern. A tool called NOVA[68] was developed to analyze power supply drop across the whole chip, using a distributed R, C and L model of the power supply rails as well as the first-level package together with estimates for the switching loads across the chip. This tool has also been used in IBM microprocessors to optimally place de-coupling capacitors to minimize the power supply noise. As power dissipation becomes a major metric for high performance designs, IBM has leveraged its expertise in fast circuit simulation techniques to perform power analysis on large function units. This has been done using ACES which due to its fast speed and large capacity (multi-million FET's), can analyze large functional units at the transistor level with an accuracy unrealized by older techniques based on simplified switch level tools.

Transistor-Level Timing

Due to the push to meet aggressive timing requirements, transistor-level timing analysis has become a necessity. IBM has leveraged its investment in the static timing tool EinsTimer by extending its capabilities to the transistor level by incorporating a transistor-level timing tools (EinsTLT) as part of the EinsTimer system. EinsTLT in turn uses the fast simulation capabilities of ACES to perform timing analysis at the transistor level and seamlessly provides this to EinsTimer. EinsTLT used its circuit topology recognition methods to add the capability to perform static timing analysis on SOI circuits, which have unique behaviors such as the floating body history effect.

Beyond accurate analysis, meeting timing goals has required optimization tools for transistor and interconnect sizing. Jiffytune [69] a dynamic tuner using a highly-sophisticated general-purpose non-linear optimization engine has been developed for optimizing critical paths and has been used successfully to optimize critical paths in IBM microprocessors. A novel approach to circuit tuning using static timing has also been recently pioneered at IBM [70], [71]. The circuit tuning work was recognized by the operations research community as part of the INFORMS award to IBM for its pervasive and innovative use of optimization techniques across the corporation.

Memory Array Design

An increasing portion of a chip's content is on-chip memory, and this is driving significant tool development. Systematic timing, noise and power analysis have been used for a long time on logic designs, and are now beginning to be employed on array designs. A key advance is the use of behavioral models in ACES, which abstracts away the details of each memory cell while keeping the transistor level description where necessary, allowing the simulation of the whole array at a level of accuracy hitherto available only for small circuits. The timing and noise abstracts generated this way can then be used in higher levels of analyses.

Package Design and Analysis

The board and MCM designs for IBM's enterprise servers continue to be among the most complex in the industry [72], with up to 29 chips, over 600 meters of wire and 4200 I/O, and CPU frequencies of over 600MHz. To perform the package design and analysis for such systems IBM has utilized a combination of internal and external tools for physical design and analysis. External tools such as Allegro from Cadence Design are used for design entry but internal tools are used for routing, timing and noise analysis. The noise tools use a novel statistical cross talk algorithm [73] which has been shown by extensive use in production designs to be much less pessimistic then traditional deterministic approaches. This algorithm includes the effect of timing variations in aggressor nets on near and far end noise.

Circuit Quality and Robustness

With the increasing use of aggressive dynamic circuits and large functional units, IBM has developed tools to guarantee that custom designs adhere to a uniform design style and to improve their robustness against process and timing variation. Einscheck is a flexible and extensible tool that performs static and dynamic checks on a custom design. It checks topology, electrical constraints, beta ratios, latch styles and signal waveforms to insure that design rules are followed. It can be customized to new technologies and design methodologies.

D - Physical Design

IBM's constant pursuit of the highest performance chip and packaging technologies has demanded repeated innovation and sound software engineering in the physical design arena. Fortunately, IBM has been able to develop highly automated design systems to support a remarkable advance in ASICs: 1) <1K gates in 1972 [74], 40K gates in 1984 [75], 3) 300K gates in 1990 [76], 4) 3.3M gates in 1994 [77], 5) 24M gates in 1999 [78] and 40M gates planned for 2000.

IBM began using automated module placement in the mid-60's and by 1972 had developed a fully automated physical design system to support the S/370 product line. This production system consisted of a set of host-based batch tools which handled cards, boards, and multi-chip modules as well as chips. Use of a common, hierarchical database allowed design details, such as I/O assignments, timing, and noise, to be passed between levels of packaging. A strict methodology was enforced by audited checking functions. The system prevented such things as running detailed wiring unless detailed placement had been previously run and checked.

In the 90's a new workstation-based design system, ChipBench, was developed as a suite of tightly-integrated tools sharing a common in-memory model of the entire design [78]. Tools operated incrementally, allowing a designer to monitor the impact on timing as a floor plan is being modified and as logic optimizations are being performed. The increase in density has led to hierarchical chip physical design and the system had to handle different design methodologies for 40 million gate ASICs and 200 million transistor microprocessors. Designers needed the flexibility to run tools out of order for early analysis, such as running global wiring to get better net delay and congestion estimates between large blocks prior to placing some of the smaller functions. A strict audited methodology could no longer be used. The technology and library descriptions were made more accessible, allowing designers to customize them when needed. Area planning helps predict the size each piece of the logic hierarchy will require. Early floor planning resolves timing and congestion problems prior to detailed logic design based on assertions. The system supports automatic floor planning, as well as manual editing to resize and reshape the blocks.

In the early 90's, IBM began a cooperative effort with the Institute for Discrete Mathematics, headed by Professor Korte at the University of Bonn, to explore the application of large scale optimization methods to the challenges of chip physical design. The result is a set of tools including placement, wiring, timing optimization, and a continuous gate and wire sizing algorithm along with capabilities for clock scheduling and optimal timing analysis with transparent latches [79]. These tools have been used extensively in IBM on many of the most demanding designs.

Placement

The early placement tool was a collection of interchange techniques, each focusing on different criteria, such as congestion, net length, and voltage drop [74]. As additional constraints were added, a simulated annealing technique was adopted, which allowed all of the constraints to be handled simultaneously in a single cost function [80]. Nets which would be optimized in a later step, such as clock trees, scan chains, and re-powering trees, were ignored during the first placement run. By the mid-80's, the delay of on-chip nets was becoming significant. Initial static timing analysis was run using rough predictions of net delays based on net type and number of pins. These timing results were used to generate the minimum and maximum capacitance constraints for the first placement run. After placement, improved net delay estimates were generated and fed back into timing analysis. The logic designers started running this placement/timing analysis iteration themselves, rather than waiting for feedback from the physical design center.

Placement programs now have to handle flat designs with over two million placeable objects, which include a mix of large macros and medium cells plus some very small cells. The PDS system, described earlier, uses a combination of min-cut techniques with multi-layer partitioning and simulated annealing. Even a hierarchical design may be placed flat with placement constraints transferred from floor planning. Specialized techniques have been developed to handle bit stacks. Another effective placement tool uses quadratic optimization and a new quadrisection algorithm, which minimizes movement instead of cut-nets [81]. A minimum cost flow approach has also been added [82]. In all cases, the placement functions are tightly coupled with timing analysis and the other optimization functions that operate at this stage of design, such as clock tree generation, scan chain reordering, circuit power level assignment, and buffer insertion [47].

For placing chips on modules, an interactive graphic tool has been developed that supports manual placement of over 130 chips on multi-chip modules and also places modules on boards [83]. An integrated set of analysis tools was used to guide the manual placement. These included static timing analysis, simultaneous switching noise, and cross talk noise [84].

Wiring

Two-layer wiring for the early bipolar chips involved global routing, vertical track assignment, horizontal line packing, followed by a cleanup maze runner [85]. Circuit density doubled in the early 80's with the addition of a third wiring layer. With this complex circuit layouts began blocking more of the wiring space. A new wiring technique was invented to handle this [83]. The global router focused on congestion, capacitance, voltage drop, and min/max timing constraints, so the detailed router could focus on pin access. The detailed router used a unique packing technique, with access from every pin to the upper wiring bays being monitored as wires were added. Priority was given to connections with poor pin access. Partial connections were routed to prevent pins from becoming blocked by the wires of other nets. Vertical packing was done first for each global routing column. The same technique was then used to route the global rows. This technique lent itself to parallel processing, and a graphic monitor was developed to display the wires as they were being generated. The cleanup maze runner was enhanced to allow rip-up and reroute of existing connections while not violating the constraints. The maze runner also divided the chip into overlapping regions to reduce memory and allow parallel processing. Long connections were wired in pieces rather than one large maze run covering the entire chip. As densities increased, so did the focus on the signal wire's impact on yield and reliability. The wiring tool was enhanced to prevent conditions known to cause manufacturing problems such as spreading wires to reduce the probability of shorts.

While the bipolar chips used a single wire width, wide wires are now used with CMOS to reduce resistance on critical nets. The width of each wire segment can be tuned to meet clock skew and electomigration targets. Critical nets can also be given a larger spacing to reduce capacitance and noise. Early noise analysis is used to guide detailed wiring's noise avoidance. The Bonn local routing is based on optimum Steiner trees and shortest path search with an interval-based routing grid data structure [78].

An important aspect of designing gate-array chips was wirability analysis [86], [87]. The amount of wiring space needed for hundreds of future designs had to be estimated as part of the gate-array image design. Extensive experiments were run to improve the wirability theories. These experiments covered the impacts of aspect ratio, additional wiring layers, perimeter versus column driver cells, and embedded SRAM macros. Every circuit layout was reviewed by a wiring tool developer to recommend pin access improvements. Since the automatic tools could not guarantee 100% wiring for the over 700 gate array designs, an interactive graphic wiring tool was developed [88]. It allowed a user to manually modify the wiring while performing physical and electrical checks. A maze runner assisted the process of adding wires.

While bipolar gate-arrays used a fixed and predefined power distribution layout, CMOS power grids are automatically customized to handle large macros and off-chip driver placement. The final power distribution can be analyzed for voltage drop, electromigration, and noise [89]. The results of the noise analysis are used to guide the placement of decoupling capacitors on the chip.

High-performance chips demand high-performance packaging and by the late 80's, modules for the IBM ES/9000 had 69 ceramic layers with nearly 3000 pins and 400 meters of wiring. For package wiring, the pins of each net are ordered

to meet timing and transmission line constraints. Each two pin connection has a minimum and maximum length along with a range of wiring layers which it could use. Clock nets are routed first and manually fine tuned before the remaining nets are wired. In addition to the length and layer constraints, the wiring tool handles cross talk noise avoidance. The early work on these packages forced design tools to handle large designs and to deal with interconnect effects such as inductance and noise avoidance that help in handling today's large high frequency SoC designs.

Logic Optimization in Physical Design

In the 70's, no logic changes were made during the physical design process, but it soon became apparent that some changes could significantly improve the results, like swapping equivalent inputs to better align pins for wiring. As the ability to verify correctness improved, more complex changes were made. Clock trees and scan chains were generated as part of the placement process. Circuit power was adjusted based on the wiring load it had to drive. As clock frequencies increased and interconnect delay became significant, clock distribution became an important issue. A variety of techniques were developed to generate delay-balanced routing of clock nets [90], [91], and to optimize the assignment of clock sinks to nets in buffered clock trees [92]. Initially, buffered clock tree generation was done using simulated annealing. As clock tree sizes grew and designer demand for fast clock tree optimization in an ASIC design methodology increased, this was replaced with by a combination of initial greedy clustering followed by iterative merging and re-partitioning of the sinks of pairs of adjacent clock nets. For high-speed microprocessors, more specialized techniques were used, based on detailed clock net analysis and including wire routing, widening, and shielding [93].

E - Manufacturability

IC manufacturing in the 50's was accomplished with the use of photomasks designed by hand with colored plastic sheets and mylar tape. The first computer aided layout design tools allowed the design of mask geometries in the form of punch cards which drove numerically controlled film cutting machines [94], [95]. Manufacturability of a design required that geometric shapes did not exceed minimum size limits, shape overlays satisfied process variation tolerances and the interconnects and devices embodied by the design accomplished the desired function. These requirements were verified by visual inspection and post-manufacturing testing.

In the mid 60's and early 70's, IBM developed the industry's first tools for interactive design of mask geometries [96], a the first design rule (DRC) and layout vs. schematic (LVS) checking tools [97], and the first hierarchical mask geometry database (GL/1)[98]. These advances allowed for IC designs to be stored, audited and verified, such that manufacturability of the design was guaranteed through verification before actual release to manufacturing.

IC manufacturing shape data preparation (DataPrep) transforms the mask layouts in the geometry database to the geometric data used to drive the lithographic patterning steps of manufacturing. Until the early 90's, DataPrep was limited to mask fracturing in which polygonal shapes were converted into mask making tool representations. During the 90's, the increasing complexity of mask and wafer fabrication and the use of aggressive lithographic patterning required new data preparation techniques, including optical proximity corrections, density effect compensations, and phase shift mask generation [99]. Thus, modern design manufacturability depends on the effectiveness of DataPrep.

These requirements in addition to the enormous growth in geometry database sizes was anticipated by the development of a hierarchical, universal and programmable shapes processor, Niagara[100]. The Niagara shapes processing engine now includes DRC, LVS, CAA (see below), DataPrep and technology migration among its many applications.

DataPrep techniques also found novel uses in the 90's. As the race to shrink critical dimensions heated up, the need to reuse IC designs with newer technologies becomes critical. The process of converting a layout to a new technology, called technology migration, involves complex layout mapping including layer generation, shape scaling and biasing and changes in layout topologies, which have been accomplished with Niagara applications. As new technologies become more dissimilar, even more sophisticated mapping techniques are necessary such as the minimum perturbation compaction method developed at IBM [101].

Design for Manufacturability

Manufacturability has long been defined by the pass/fail criteria of DRC and LVS checking, even though it was soon realized that the actual fraction of good parts, or yield, depends on the detailed photomask design. Stapper [102] at IBM pioneered the techniques to predict the yield of a part before manufacturing by Critical Area Analysis (CAA) of the design masks. The work of Stapper on yield enhancement of memory arrays [103] by the use of redundant layouts also demonstrated that design and layout can strongly improve IC yields. The concept that manufacturability can be measured and improved through design practices is called design for manufacturability (DFM). Initially DFM was applied in IBM with DataPrep techniques using Niagara [104]. DFM requires a robust yield prediction capability, and therefore the CAA techniques at IBM have been enhanced to become a massively distributed, full-chip yield prediction tool [105]. With the aide of the yield prediction capability in CAA, a novel yield-aware maze routing technique has been developed [106]. Further work on yield-aware routing and compaction techniques have shown as much as 20% combined yield increases are possible. The concept of design for manufacturability has also been demonstrated through the manufacturing of a yield-enhanced PowerPC 750 microprocessor [107].

The trends of pricing and cost pressures, increased pace of new, increasingly complex and aggressive technologies combined with shortened design cycles are setting a new pace for the future of manufacturability and EDA tools. These pressures are now demanding early estimates of yield, cost and reliability. In addition, traditional tools such as synthesis, placement and wiring are becoming increasingly yield aware. Furthermore, technology migration tools will increasingly be used to optimize layouts for manufacturability. Clearly, manufacturability is a critical metric for designers and is being incorporated throughout the design process.

	Table 2 - Design Tool Milestones in IBM	
	Logic Design	
1974	First use of heuristic PLA minimization	MINI [37]
1979	First production use of block-oriented timing	TA [29]
1982	First production use of transformation-based logic synthesis	LSS [39]
1983	First publication of redundancy removal in logic synthesis	LSS [40]
1984	Seminal publication on multilevel logic synthesis	YLE [38]
1986	First use of incremental timing in logic synthesis	SlackHoe[32]
1986	Global Flow analysis in logic synthesis	LSS[41]
1990	Common delay calculator for all design tools	DCL
1995	Demonstration of combined physical/logical optimizations	BooleDozer [47]
1997	Gain-based synthesis used in production logic synthesis	BooleDozer[44]
1998	Wavefront technology mapping used in production	BooleDozed[45]
	Transistor-Level Design	
1967	Graph representation of circuit topology	[50]
1971	Sparse tableau approach used for circuit simulation	ASTAP[51]
1974	Partial Element Equivalent Circuit used in parasitic extraction	COSMIC[61]
1975	Modified Nodal Analysis (MNA) used for circuit simulation	AS/X[52]
1978	Automatic layout checking in Universal Shapes Checker	USC[97]
1991	Production use of piece-wise constant/linear FET models	SPECS/ACES[57], [58]
1994	Statistical noise estimation for package design	Sxtalk[73]
1996	Production use of efficient automatic circuit tuning	Jiffytune, Einstuner[69]
1996	Full chip power distribution analysis	NOVA[68]
1997	First static noise analysis for digital circuits	Harmony[66]
1998	Full chip global parasitic extraction Including Inductance	3DX[68]
1998	Comprehensive custom logic circuit checking	Einscheck
	Physical Design	
1972	Integrated Physical Design System for chips, modules, cards, and boards	[74], [84], [85], [88]
1981	Simulated Annealing applied to placement in production	MCPlace[80]
1983	First four level metal chip wiring	XAWire[83]
1989	First production cross talk avoidance wiring	MCMRouter
	Physical Design System to support IBM's high-performance technology	
1984	- 40K Gates	[75]
1990	- 300K gates	[76]
1994	- 3.3M gates	[77]
1999	- 24M gates	[78]
	Release to Manufacturing	
1966	Hierarchical mask data model: Graphics Language/One	GL/1[95], [98]
1972	Interactive, graphical mask editor	IGS 1130[96]
1974	DRC and LVS using Universal Shapes Checker	USC[97]
1980	Yield Prediction from mask data using Critical Area Analysis	CAA[43]
1993	Hierarchical manufacturing data preparation with OPC	Niagara[99]
1998	First design-for-manufacturing enhanced PowerPC 750 microprocessor	[112]

F - Manufacturing Test

Because of the large volume of chips being designed, IBM, early on, pursued and adopted Automatic Test Pattern Generation (ATPG) methods based on structure-oriented test methods. It was found that ATPG would be more practical if internal registers in complex designs are made accessible by a dedicated scan approach [108]. In the 70's, Ed Eichelberger proposed a revolutionary approach, named Level Sensitive Scan Design (LSSD) [109], [110] that makes timing-robust correct-by-construction test programs for a wide range of products and test equipment parameters possible. LSSD quickly became IBM's methodology of choice [111]. ATPG algorithm advances, like the PODEM algorithm [112] that succeeded the well-known D-Algorithm [113], became significant drivers for a practical set of test generation tools that still is used today. Another tool innovation was the development of design rules checking software that automatically analyzes a design for compliance with the LSSD architecture requirements [114] prior to chip release and test generation.

An LSSD-based logic Built-In Self-Test (BIST) architecture called STUMPS, the prototype of most logic BIST schemes used in the industry, was introduced in the 80's [115]. New fault simulation technologies that evaluate multiple test patterns in parallel using compiled code [116], [117] and event-driven interpreted methods (Parallel Pattern Single Fault Propagate, PPSFP) [118] were pioneered for BIST. In a related development, IBM engineers also created a innovative chip-level test method called Weighted Random Pattern (WRP) test [119], which uses encoded test patterns to improve the memory utilization of the test equipment.

At-speed test capabilities were devised for scan and LSSD early on [120], [121]. In the early 80's a simple, very pragmatic method to convert static LSSD tests into timed tests [122] was introduced for chip and module testing. A comprehensive automatic delay test generation system [123] for production use followed in the early 90's. Among the key innovations for this system are the transition fault model [124], a small delay fault model and simulator [125], and a timing tool to determine the appropriate test for an at-speed test [123]. Another significant technical contribution of the time pioneered robust delay test criteria for path delay faults [126].

Embedded Memory and Macro Test

Many embedded memories are designed for density, leading to very specific failure modes and, sometimes, the need for repair. Such memories are best tested and diagnosed with specialized highly regular algorithms. IBM pioneered an approach and supporting tools for accessing embedded memories or other macros, such as embedded processors, through surrounding logic [127] for testing. Early on, it was suggested that memory tests, due to their simple and regular nature, could be implemented in BIST hardware right next to a memory macro on a chip or module [128]. This idea evolved into more sophisticated architectures with flexible test algorithms that can be tuned to the particular memory configuration under test on a chip [129]. ABIST today is used in virtually all of IBM's chips.

Boundary Scan

Boundary Scan uses special scan cells associated with the chip pins [130]. IBM has used LSSD-based boundary scan cells and associated test methodologies in practice at least since the early 80's [131]. It is also worth noting that the chips described in [131] contained an On Chip Monitor (OCM) port that standardizes access and control of all boundary scan, internal scan, logic BIST, ABIST, and other test resources.

IBM today extensively uses boundary scan in innovative I/O test strategies for IC manufacturing test. IBM's unique I/O wrap test features support a very cost-effective Reduced Pin Count Test (RPCT) approach for wafer sort, where only a subset of the chip I/Os needs to be contacted [132], [133]. The unique boundary scan implementation facilitates at-speed wrap-around testing at wafer sort and DC parametric testing of the I/Os at final (package-level) test. Some very high-performance products use LSSD-based boundary scan at the package level to support at-speed interconnect testing [134].

TestBench

TestBench, today's workstation-based suite of IBM test tools, includes a Design-For-Test Synthesis (DFTS) system that automates the insertion of internal and boundary scan structures, the configuration and insertion of logic BIST and memory array BIST structures, the configuration and insertion of IEEE 1149.1 TAP controllers, and the construction of a chip top shell. The design rules checking and testability analysis tools in TestBench are coupled to

a graphical netlist browse and edit utility, which automatically extracts relevant pieces of the netlist and creates an annotated interactive schematic snapshot for the user. The user can then traverse the netlist, apply simulation and test generation commands, and even edit part of the circuit for experimentation purposes. The ATPG, fault simulation, and logic diagnostic applications in TestBench include a unique fault model concept called pattern faults [135] that complements the standard fault models in TestBench (including stuck-at faults, transition faults, path delay faults, I/O faults, and IDDq faults). Pattern faults are a convenient method to define specific local test conditions for such failure modes in the netlist that cannot be easily derived from any of the other fault models (e.g., shorts). TestBench uses a sophisticated Multiple Test Mode (MTM) architecture that allows a user to define and keep track of different test setup conditions (e.g., for logic, embedded memories, I/Os, and any other special building blocks), to monitor the associated local and global fault coverage statistics, and to integrate the different tests into a global flow.

ASIC and Microprocessor Test

IBM's ASIC design sign-off flow is unique in the industry. By using a ASIC Sign-Off Kit (ASOK) that includes robust DFTS and DFT structure verification tools, customers are completely relieved from having to generate chip manufacturing tests themselves [136]. A small manufacturing support team, taking full advantage of the LSSD, ABIST, macro test, Reduced Pin Count Test, and MTM features in TestBench, performs all ATPG runs, even for the most complex ASICs with embedded processors.

Processor design today produces very complex custom CMOS using a combination of HDL Synthesis and hand-optimized transistor-level design techniques. Processor design teams are very concerned about achieving gigahertz performance and tight layouts by applying very aggressive circuit-level design "tricks" (like pass-gate logic, dynamic logic, self-timed or self-resetting logic, and complex multi-phase clocking styles). Although TestBench can accommodate some transistor-level models it is much more practical to derive a suitable gate-level model from the transistor-level design and let the DFT/ATPG tools operate on this derived model. The TestBench tool suite includes a sophisticated model extraction tool called Gatemaker [137] for this purpose.

One key problem with clock frequencies approaching the GHz level is the lack of affordable test equipment that can handle such high frequencies. IBM has over the years pioneered and developed a number of On-Product-Clock Generation (OPCG) techniques for at-speed test [138]. With OPCG, the tester only needs to send a reference clock, optionally multiplied by an on-chip Phase-Locked Loop (PLL), and the test timing edges are generated on the chip under test itself. Some tests, like the measurement of embedded memory access, setup, and hold times, use programmable on-chip delay lines and associated calibration techniques for higher-resolution signal edge placement [138].

Logic Diagnostics

A significant industry trend is that semiconductor processes are being brought up with logic products, making logic diagnostics and failure analysis increasingly important for early process learning and yield improvement. The simulation-based diagnostics first pioneered by IBM in the Tester Independent Chip Diagnostics System (TICDS) in the 80's [139] are a significant improvement over traditional dictionary-based methods. The TICDS approach can also be used for logic BIST and WRP [140]. To enable logic BIST and WRP diagnostics, IBM pioneered a simple method to dump the contents of all scan cells out to the tester for later detailed analysis after a signature mismatch indicates defect detection. TestBench includes a number of advanced graphical visualization tools, wave-form display tools, circuit trace, simulation, and analysis tools for logic diagnostics.

Defect Based Testing

The purpose of test is to find and diagnose defects. Test generation and diagnostic tools, on the other hand, use fault models. The ability to establish a strong correlation between defect levels and fault models is vital for assuring high product quality and to guide the generation of more efficient and effective tests. IBM has a rich tradition of defect analysis and fault modeling that includes work on modeling the relationships between defect levels, test coverage and product quality [141], [142], the modeling of defects and yield as it relates to memory redundancy and repair [143], and the use of circuit level simulation to help evaluate fault models and testability of logic library elements [144]. Another important contribution is the use of critical area concepts to model the sensitivity of layout elements to defects of different sizes [145]. Finally, IBM has always complemented the modeling and theoretical analysis with empirical test effectiveness and failure analysis work [146].

	Table 3 - Manufacturing Test Milestones in IBM	
1966	Invention of D-Algorithm for pattern generation	[113]
1973	First use of Level Sensitive Scan Design (LSSD)	[109]
1976	Early Memory BIST and Embedded Macro Test	[127], [128]
1981	PODEM Algorithm (first with implicit enumeration of decisions)	[112]
1981	First simulation-based logic diagnostic system	[124]
1982	Early Boundary Scan and On-chip Test Interface	[130], [139]
1982	First widely used scan-based Logic BIST(STUMPS)	[115]
1985	First Industrial Parallel Pattern Fault Simulators	[116], [117], [118]
1985	Robust Path Delay and Transition Fault Models	[124]
1987	Production use of Weighted Random Pattern Test	[119]
1989	Introduction of Reduced Pin Count Test	[132], [133]
1990	Early Industrial Scan-Based AC Delay testing	[122], [123]
1996	Pattern Fault model for non-classical defects	[135]

III - Future

With so many new technology, circuit and architectural developments being explored, no one can accurately predict the requirements for a future design system. But, it seems certain that in the near future the embedded processor and System-on-a-Chip (SoC) markets will continue to explode as processing power and increased integration is delivered to every segment of the electronics industry, including consumer (set-top boxes, game machines), wireless (cellular handsets), wired (Internet infrastructure), pervasive (printers, GPS), storage (SAN, RAID) and server (Internet infrastructure). Continued advances in technology are enabling the design of larger and denser chips, but are out pacing designer productivity. Multi-chip packages and single multi-technology chips, with a growing diversity of technology (RF, Analog, FLASH, FPGA, CMOS, embedded-DRAM, SiGe, …), will be deployed to insure the lowest manufacturing cost for each technology. Embedded RISC processor performance will surpass the 1GHz clock rate utilizing embedded DRAM for L2/L3 cache and re-configurable instruction sets. Time-to-market pressures will drive integration at the expense of both area and performance optimization for many applications. The time to verify new and unproven logic will continue to be the critical path for getting a product to market. The need for extended battery life and low power requirements will continue to be a challenge as technologies continue to lower the threshold voltage to combat the increased clock frequencies and associated increases in power dissipation.

Successful companies in the future will require the capability to rapidly develop complex hardware and software systems by utilizing and customizing existing components as well as generating and designing new high-performance components, all with an extremely productive and predictable process. It will be necessary and possible to implement these systems with a productivity much greater than today, despite the design problem becoming more complex due to the diversity of chip components. Figure 1 depicts a vision of SoC design in the future. The sections below describe the requirements and initial thinking for the four major elements:

- System Specification and Verification
- Design Mapping
- Design Implementation and Optimization
- Release to Manufacturing and Test

A - System Specification and Verification

A SoC designer in the future will begin by developing a functional specification that captures the desired behavior at the highest possible level of abstraction that can be simulated, along with a set of constraints on the many design parameters for the specific product. The specification format needs to be as unrestrictive as possible, but will most-likely be based on interconnecting a set of functional units or components that can be drawn from a library of parameterized representations. More important than the format is a sound and simple semantic foundation that is appropriate to both the designer's decision process and the verification tools. As this specification is being

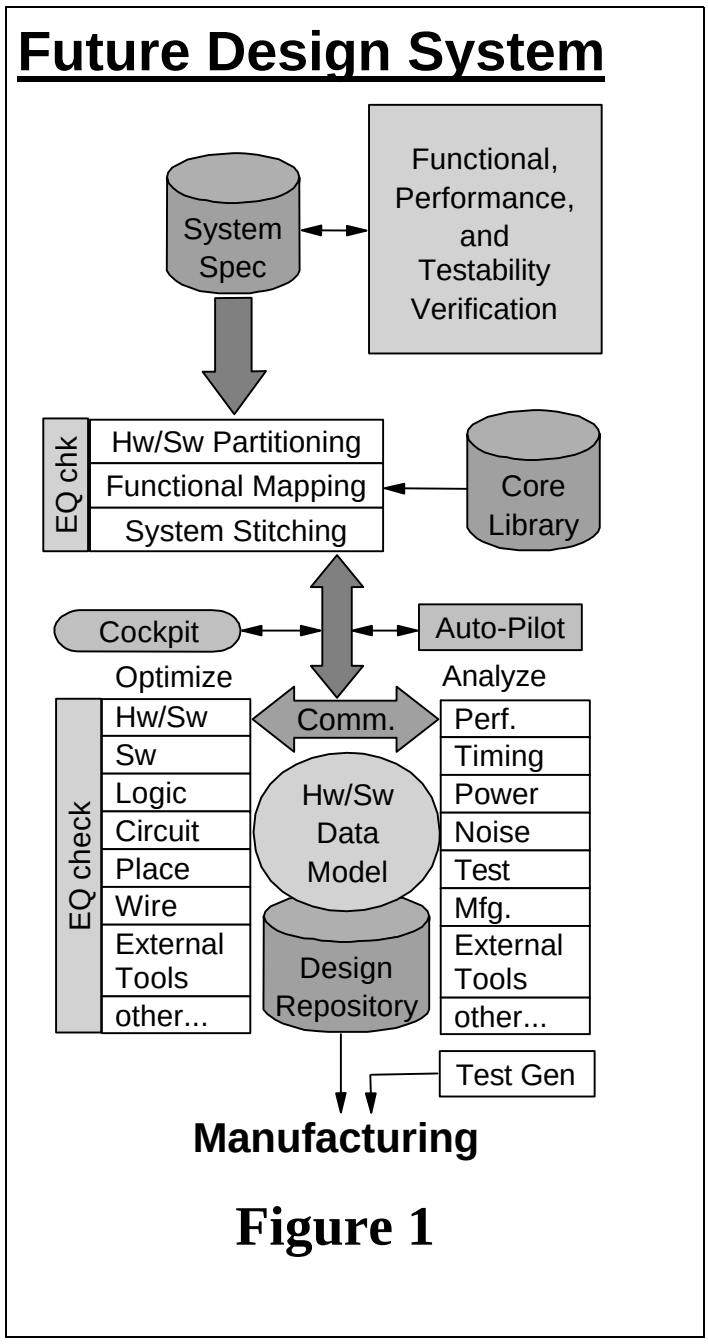

Figure 1

developed, a suite of verification tools will be used to confirm not only the correct behavior, but also the correct values for other design parameters, such as power dissipation, performance, or cost. Ideally, the SoC designer with this high-level specification will be able to confirm the product's desired behavior through functional verification, and determine an early estimate of the values of the key design parameters, such as cost, power, performance and testability, through a set of design parameter verification tools. As the design progresses from specification to manufacturing, bounds for the design parameters will converge to accurate predictions of the final product's performance. New equivalence checking tools will confirm that the specified function is preserved as the design is transformed into a final manufacturable implementation.

SoC functional verification will require significant advances in simulators targeted at specific component types, such as RF, analog, digital and mechanical, plus a standard environment for coupling these tools together without sacrificing efficiency. Some relief will come from new hardware accelerators and emulators as the race for improved capacity continues. Formal verification will advance and play an even larger role in proving of important properties of the specification, raising confidence in it's "correctness". There is work on increasing tool speed and capacity and also on improving the user interfaces to make model checking more attractive for designers to easily and exhaustively explore their design space. Combining methods of simulation with formal verification is a promising area of research in IBM with the potential of applying high-speed simulation to drive a design into 'interesting state conditions' and then continue with formal, exhaustive evaluation methods from that point on. Finally, a methodology is needed to capitalize on the body of prior verification results when an existing component is reused. It must not be necessary to reverify the entire system, as if it were all new.

Design parameter verification will require calibrated estimation tools that produce a range of values to check against the design constraints. The calibration is with the software and hardware implementation tools that follow. They must be able to reliably deliver results within the estimated ranges to avoid time consuming redesign loops. While many research breakthroughs may be required to achieve these capabilities, the potentially dramatic improvement in productivity certainly justifies their pursuit.

Equivalence Verification

Regardless of the level of automation following System Specification and Verification, it is essential that equivalence be preserved between the specification and the implementation as it progresses through Design Mapping and Implementation/Optimization to final realization. Rerunning the specification tests will provide some confidence, but what is needed is new high-level equivalence checking tools that can be used in this context and also after each of the later Mapping and Implementation/Optimization steps. A verified architectural model can be used to confirm component and system behavior and performance, to develop expanded test cases, to support pre-silicon software development, and to be used as a pre-silicon marketing vehicle.

B - Design Mapping

Design Mapping refers to that part of the process where critical early design decisions and tradeoffs are made that determine the fundamental nature of the end product. That is, the functions specified above must be mapped to generic hardware and software components, which in turn must be translated to specific hardware that can be manufactured.

Hardware-Software Partitioning

As the specification is being developed, the SoC designer will create an implementation "architecture" as a network of generic hardware and software components. This assignment of function to hardware and software will largely determine the values of all design parameters and will be done by expert designers with the assistance of high-level analysis and estimation tools together with a set of optimization tools to help meet the specified design constraints. Software components are bound to hardware components that establish their performance, power, and other parameters. At this point the generic components can be positioned in a system floorplan to guide later analysis.

Functional Mapping

Functional mapping is similar to the technology mapping step in logic synthesis, but for larger elements. This step will use algorithms for mapping a technology independent system specification onto a library of specific hardware components that can realize the system. The mapping may be to a single component or a combination of components. This step also uses algorithms and tools for optimizing the mapping to achieve the cost, area, timing and other design constraints.

System Stitching

In addition to selecting components, appropriate interconnect, converters and other infrastructure elements will need to be added to complete the implementation. This will be accomplished with a set of tools that understand the components and their requirements for assembly to be able to be able to put them together quickly and efficiently. System stitching will take advantage of predefined architectures to quickly produce efficient implementations. Following this step the refined floorplan should enable more accurate estimates for the design parameters.

C - Design Implementation and Optimization

During the mapping process, a specific hardware component may not be available and a new implementation will be required. The required implementation could use a range of design styles including ASIC, custom and semi-custom, depending on the specific requirements of its environment. Even for those components which are available, some may exist as logic-only (i.e. "soft") components which will be open for further optimization during implementation. In the final assembly of the hardware and software, newly implemented, logic-only, and library (i.e. "hard") components will be incorporated along with the needed interconnect adapters and converters. Hardware component placement will be finalized on chip and the required power and clocking circuitry will be added. In addition, there are opportunities for optimization as the design progresses towards manufacturing. The design system to support these implementation and optimization steps will require simultaneous use of the complete set of analysis functions and comprehensive optimizations capabilities, all cooperating on a shared design representation. This requires a new design system architecture.

New Design System Architecture

Earlier, Section II.B. describes an evolution of the IBM design process through the 90's towards an integration of the previously separate logical and physical design capabilities. While design times have been improved with placement-directed synthesis, more is required and can be achieved. The step by step integration of timing with

synthesis and then with placement has been generalized into a vision of a high performance SoC design methodology enabled by new application modules integrated through a new design system architecture. In IBM, tools from the logical and physical environments are being redesigned following the guidelines of the Unified Physical Design and Synthesis (UPS) architecture. A unified development environment is being established for migrating the existing capabilities and developing new ones, as more of the critical design methodology segments are addressed, all fitting within this new architecture. To be specific, the objectives set forth at the start of the UPS development were to eliminate the duplication of environments between synthesis and physical design, making it much easier for designers to do a full synthesis and placement run from one system, and to improve the turn-around time of one iteration through the design closure loop by employing tightly-integrated, smaller algorithmic steps, early bailout of unfruitful corners of the design space, elimination of duplicate timing environments, and unnecessary database transfers. The resulting system has the following attributes:

- A single execution environment, called Nutshell, which enables dynamic loading and binding of modular functions at runtime. This allows the execution to be dynamically configured and easily extended to address specific design tasks.
- A single runtime data model, the Integrated Data Model (IDM), which provides a generalized callback mechanism to enable function interoperability and incremental processing.
- A common electrical subsystem to support of parasitic modeling, model reduction and delay calculation.
- A common user interface and common handling of application parameters.
- Rich sets of granular, reusable functions covering the following areas:
  - logic optimization (synthesis, design-for-test)
  - physical optimization (placement, global wiring, detailed wiring)
  - clock-scan optimization
  - gate sizing, buffer insertion, wire sizing, layer assignment, Steiner estimation, etc.
  - incremental analysis (timing, noise, power, extraction, checking, etc.)
  - logic and physical editors and browsers
- All functions, including IDM, are supplied with command language bindings to enable easy customization.

With UPS as the base, we must shift our focus towards the future where the design system must allow designers to rapidly assess the status of the design and apply a wide variety of incremental optimizations to drive the design toward the acceptable criteria.

Design Analysis

Traditionally, designers have considered relatively few factors, perhaps only performance and area. However, even today's designer is faced with a large and growing number of constraints including battery life, weight, noise, and yield. To support automatic or manual optimization, the design system must provide incremental analysis of many design parameters [147]. Before describing some examples, we discuss some methods for efficient computation.

Demand-driven analysis is one aspect of this, in which we start by querying for a particular analysis result and then recursively compute the necessary information to answer the query. For example, if in a static timing analyzer we are interested in clock skew, we would need to compute the arrival times at all clock inputs of latches, which would in turn require computation of the arrival times and delays in the cones of logic feeding these clock inputs. This demand-driven computation can cross analysis domain boundaries, so the delay calculator might in turn ask for electrical information which would in turn ask for estimated routing information, etc.

A second way to reduce analysis computation cost is to compute answers with only the level of accuracy required. For example, if the analysis is to verify compliance of a physical design constraint, the requirement may be for a simple yes or no answer. An initial fast, low accuracy analysis can be performed to identify regions of the design which clearly violate constraints, and those which clearly do not. A more expensive and accurate analysis can then be performed only on the uncertain regions. An analysis domain executive can control the local accuracy level used, hiding local accuracy selection from the requester (the designer or another tool). Sensitivity information can be used to determine the level of accuracy needed for intermediate analysis results (e.g., electrical parameters) contributing to the final answer of interest (e.g., timing slacks).

System Performance: Performance modeling is commonplace in the design of processors and certain mature application areas, such as analog-digital converters, filters, and switches. But such predictive models are the result of an expensive and lengthy development process. Part of the SoC development process will require the ability to rapidly assemble accurate performance estimators to provide feedback during system-level partitioning and optimization. Some required performance measurements are: average and peak CPU utilization, worst case critical interrupt latency, average and peak bus utilization, measures of the interaction between bus utilization and CPU utilization, maximum depths in hardware and software queues, RTOS overhead.

Timing: The future will pose many new challenges to static timing analysis. Larger chips and higher clock frequencies will require increased use of asynchronous interfaces. New circuit families, which use pass-gates and reduce the ability to isolate the delay calculation of consecutive gates. Limited swing differential signals used to reduce power consumption can also place new burdens on accurate delay calculation. Analog components like PLLs, whose "delays" depend on external paths [148], impose new interdependencies between timing analysis and delay calculation. Also, the acyclic nature of the timing graph is compromised by interactions which are not explicit in the netlist, such as capacitive and inductive coupling between wires, and local voltage and temperature variations.

Power: Power is an increasingly important product consideration, both for battery powered applications and high-end products constrained by heat density. Accurate system-level abstractions that predict power consumption under varying usage scenarios will be essential. New hardware components are being designed using new circuit styles with multiple voltages and thresholds all to reduce power dissipation. It is important that these components also be characterized so that they can be exploited by logic synthesis. While simulation results give the most accurate switching factor information, this information is difficult to obtain incrementally during design optimization and new hybrid methods must be developed. Clock networks are a major contributor to chip power dissipation requiring careful layout of clock gating [149]. Leakage power is increasing with shorter transistor channel lengths. Pass-gate synthesis may help reduce power since there is no short circuit during the gate transition.

Noise: As advancing technology enable faster chips, noise analysis will become more commonplace in SoC design. Furthermore, noise will have an increasing effect on timing and timing analysis. This will result in a tight integration of timing and noise analysis. Noise effects will be considered early in the design flow requiring system-level estimation methods that can operate even with partial layout information. The combination of digital and analog or RF components will impose new demands on noise analysis. Instead of separate analog and digital analysis used today, tomorrow's SoC designs will require rigorous analysis of substrate noise, and the temporal and frequency domain effects of combining digital and analog noise sources. New tool capabilities will have to be developed to tackle these complex problems

Manufacturability: Manufacturability will become a more important consideration, especially for high-performance products, and will demand early estimates of yield, cost, and reliability, factors that at the same time will become harder to predict. Simultaneously, post-design data manipulation will expand to meet the needs of more complex semiconductor processes. Growing design complexity combined with the increased importance of previously second-order effects and a reduced willingness to accept "guard-banding" will cause a rapid increase in the time required for design analysis. New analysis algorithms, such as reduced-order interconnect modeling, have helped to control this growth but the key is to do only enough computation to obtain the required answer.

Design Optimization

Additional sets of optimization modules must also be provided to help the designer modify a design to satisfy the large set of design constraints. Examples of these modules are described below.

Hardware-Software Tradeoffs: Probably the most important and difficult optimization is the assignment of function to hardware and software. There are four possible implementation modalities for any given function: 1) fixed hardware, 2) reconfigurable hardware, 3) programmable hardware or 4) software. The objective is to find the "best" implementation mode for each function such that the system meets all design constraints and minimizes several other functions such as cost, power and delay. Performance considerations might dictate that a dedicated hardware solution be used, while a set of multiple mutually-exclusive algorithms might require reconfigurable hardware for the best tradeoff between area and performance. For reconfigurable hardware, it might be necessary to store and reload

configuration data, and this reconfiguration time must be taken into account when choosing an implementation. Occasionally, acceleration of specific functions is needed and special instructions may be added and supported by special-purpose hardware units. Finally, software may be used where flexibility is of paramount importance or when performance is not critical.

The future design system must support the designer in determining the modality for each function specified by:

- Interacting with the designer in selecting an implementation
- Automatically choose an implementation mode
  - Generate an instruction set for the processor
  - Generate a function library for some identified software components
  - Generate RTL specification for a new hardware component
  - Generate code for the software components
- Using other analysis and optimization modules complete the selected component implementation

Software Optimization: Both electronic design automation and software technology are well-established disciplines with their own methods and infrastructures. For software, this includes such things as programming languages, compilers, runtime environments, function libraries, and debuggers. The advent of hardware-software co-design will cause some merging between these two disciplines. Without integration, the goal of improving product time to market cannot be achieved. A future problem is to determine what sorts of communications there must be and the best way of achieving commonality. One clear challenge for DSP-like designs is to be able to generate software tools (compilers, etc.) for the DSP instruction set which has been specified by the designer.

Test Optimization: The new challenge for the DFT developers is to deal with chips containing not just a million gates but over a million flip-flops. At the same time, the mapping of high-level designs to physical implementations is becoming more and more sensitive to physical design aspects. In such an environment, the current practice of post-synthesis DFT insertion and DFT debug becomes awkward. We therefore see a strong trend toward moving the DFT insertion and design correction further up into the pre-synthesis domain. This requires both the development of new high-level DFT insertion/analysis/correction tools and tight integration with the other steps of the front-end design process. The physical design planning and timing tools must consider the inserted test structures as an integral part of the design to be implemented. The functional verification and timing analysis tools must be made aware of how the increasingly complex test structures can be disabled to force the design into the functional mode of operation. And, the test structures themselves are getting so complex and function-rich [150] that the different test modes must be verified for functional correctness, robust timing, and interaction with the functional logic.

For example, in the gigahertz performance domain, any embedded test structures for delay test must be designed for performance. An ABIST engine for test and characterization of embedded memory array timing, for example, may actually have to run faster than system cycle time. The distributions of scan control and clock signals across a large chip, as well as the partitioning and stitching of scan chains, must take placement and timing into account as never before. The DFT tools may have to automatically add re-timing and pipelining elements and automatically generate appropriate timing, design planning, and synthesis constraints.

Logic and Circuit Optimization: Besides the challenges raised by test, logic and/or circuit optimization will be continually challenged by the increasingly detailed needs of place and wire (see next item), ultimately carrying into the transistor domain with transistor level synthesis, circuit tuning, and other similar capabilities. In addition, at least logic optimization will need to address the growing proliferation of circuit styles (e.g. static, dynamic, transistor level, gain-based, FPGA) and not only synthesize within each of them but, to be most effective, should be able to choose among them. In addition, synthesis will need to support more exotic logic such as asynchronous and analog.

Place and Wire Optimizations: Physical design will continue to be driven by timing, yield, and reliability. First, the accuracy of measuring inductance, crosstalk, power distribution, noise, temperature, and similar factors needs to improve and be properly accounted for in the timing, yield, and reliability analysis. Next, the physical design tools need to incrementally use the measurement tools to reduce the impact of these factors. This can involve such things

as fine tuning the width and spacing of signal and power wires or adjusting placement to prevent thermal hot spots. These factors also need to be predicted during the early planning stages, with the later stages constrained by these predictions to prevent surprises late in the design schedule. This all needs to be done while reducing the design time as chips exceed two hundred million transistors.

D - Release to Manufacturing and Test

At the end of this process, the design will be released to manufacturing with accurate estimations of the manufacturabilty, testability, yield, and reliability along with the data for production testing. With the growth of SoC complexity, the supporting test methodology will become more complex and multifaceted. More test function will be embedded into each chip, becoming a part of the normal design process and enabling a new level of cooperation between the embedded test functions and the external test equipment and test environment.

The Automatic Test Equipment (ATE) industry is making large strides towards much lower cost-per-pin while increasing the ATE functionality. The DFT and ATPG tools today largely ignore the more powerful capabilities of the ATE. Logic scan-based test data, for example, typically are generated in a way that makes it more or less impossible to exploit the data compression features available on some modern ATE. And, the ATPG tools often have no understanding of the specific timing features on the ATE. We foresee that future DFT and ATPG tools will be more ATE-aware and tests will be generated and validated in the context of the circuit under test and the ATE.

Just as DFT and ATPG tools are expected to become more aware of the ATE, we anticipate that the ATE will have to become more aware of the embedded test support functions. For example, we already have micro-coded ABIST engines [151] with a high degree of algorithm-level, timing-level, and diagnostic programmability. That is, the embedded test functions are beginning to have software content. Today, this functionality is not directly visible to, and accessible by, the test engineer running the ATE. We expect that in the future highly programmable and flexible test, measurement, and instrumentation functions inside the product under test will be treated as natural extensions of the external ATE functions. This requires the definition of architecture standards, data models, and interface standards that allow the embedded test functions to be automatically integrated into the ATE software. We expect similar integration between embedded test functions and the software in lab-debug stations for functional debug. Likewise, we anticipate that many of the embedded test features will be remotely accessible through industry-standard physical and logical test interface standards.

IV - Summary

Throughout the decades of continuos advances in semiconductor technology there have always been concerns about the ability of design automation tools to keep pace. In IBM there certainly have been challenging periods, but looking back, it is remarkable that critical advances in EDA have occurred and allowed IBM products to be developed on ever shorter schedules with smaller design teams. In the 80's, a series of key decisions about design practices enabled true RT-level design, and the development of many innovative tools including cycle simulation, simulation accelerators, Boolean equivalence checking, static-timing analysis, production-quality logic synthesis, and automatic test analysis and pattern generation. These new facilities combined with precise circuit analysis tools and full layout automation and checking for gate-array chips and complex packages gave IBM a strong design capability in 80's and provided a solid foundation for the emerging CMOS technology.

CMOS gave rise to ever larger and denser standard-cell chips that began to displace bipolar technology in workstation and midrange products. With the help of custom design techniques even the S/390 machines moved to CMOS. Building on work of AS/400 and RS/6000 designers, a team from S/390 and the Research Division led the development of a predictable and productive custom-design methodology and drove the development of a suite of new tools. The Power design team provided further refinements as they began work on the gigahertz Power 4 chip. New formal verification methods along with dramatic capacity increases for simulators and accelerators were all needed. New transistor-level tools were created for timing, power, signal integrity, noise, synthesis, layout and checking. The growth in importance of wire delays forced a tight integration of not only logic and physical design tools, but most analysis tools. At the same time IBM's began marketing very large and high-performance ASICs commercially. This required an expansion of IBM's methodology and tools to support 40 M-gate chips with 27 ps

gates and 7 layers of copper interconnect. Design planning and high-level synthesis were developed to raise the level of design.

Once again IBM's design system evolved to support leading-edge processor and ASIC designs with competitive schedules. Looking to the future there are many difficult challenges ahead - most without clear solutions. But we see a convergence of the previous experience and tool capabilities into a tightly-integrated design system that will enable designers to rapidly translate high-level functional specifications into an architecture of software and hardware components and efficiently realize a high-performance system-on-chip implementation.

Acknowledgments: The authors want to thank the following people for their extra effort in helping prepare this paper: Bhavnal Agrawal, Paul Bassett, Daniel Brand, Reinaldo Bergamaschi, John Cohn, Tony Drumm, Amir Farrahi, Prabhakar Kudva, Andreas Kuehlmann, Bill Lee, David Kung, Hao Ming Huang, Jaime Moreno, Dave Nelson, Dan Ostapko**,** Saila Ponnapalli, John Parisi**,** Ruchir Puri, Lakshmi Reddy, Vic Rodriguez, Al Ruehli, Jeff Staten, Leon Stok, Athar Tayyab, and Bruce Wile. In addition, we salute the many developers and designers, referenced and not referenced, who have contributed to IBM's EDA systems over the years.

References

[1] M. Monachino, “Design Verification System For Large-Scale LSI Designs”, IBM J. Res. and Dev., Vol. 26, No.1, January 1982, pp. 89-99.

[2] C. Logan, “Directions in Multiprocessor Verification”, Intl. Phoenix Conf. On Comp. And Comm. 1995, pp. 25-33.

[3] L. I. Maissel, H. Ofek, “Hardware design and description languages in IBM”, IBM J. Res. Develop., Vol. 28, No. 5. September 1984, pp. 557-563.

[4] P.W. Case, H.H. Graff, M. Kloomak, “The Recording Checking and Printing of Logic Diagrams”, Proceedings of the Eastern Joint Computer Conference, Philadelphia, PA, 1958, pp. 108-118.

[5] D.F. Ackerman et al., “Simulation of IBM Enterprise System/9000 Models 820 and 900”, IBM J. Res. Develop., Vol. 36, No. 4, July 1992, pp. 751-764.

[6] W. Roesner, “A Mixed Level Simulation System for VLSI Logic Designs”, Proc. of COMPEURO 87, May 1987, pp. 196-199.

[7] S. Wimer. R. Pinter and J. Feldman, “Optimal Chaining of CMOS Transistors in a Functional Cell”, International Conference on Computer Aided Design pp. 66-69, 1986

[8] B. Wile et al., “Functional Verification of the CMOS S/390 Parallel Enterprise Server G4 system”, IBM J. Res. Develop., Vol. 41, No. 4/5, July/Sept. 1997, pp. 549-566

[9] B. Keller et al., “The Compiled Logic Simulator”, IEEE Design & Test of Computers, March 1991, pp. 21-34.

[10] T. Burggraff, A. Love, R. Malm, and A. Rudy, “The IBM Los Gatos logic simulation machine hardware”, in Proc. IEEE Int. Conf. on Computer Design: VLSI in Computers, Oct. 1983.

[11] M. Denneau et al., “ The Yorktown Simulation Engine”, Proc. of the 19th Design Automation Conference, 1982.

[12] D. Beece et al., “The IBM Engineering Verification Engine”, Proc. of the 25th Design Automation Conference, 1988, pp. 218-224.

[13] C.H. Feistel et. al, “Structured System Design and Verification”, IBM RISC System/6000 Technology, 1990, pp. 86-91. IBM SA23-2619.

[14] EETimes, November 23, 1998, Issue 1036, “Two machines seen as paving the way for next-generation microprocessors — Quickturn boosts emulation to 20m gates”.

[15] A. Aharon et al., “Verification of the IBM RISC System/6000 by a dynamic biased pseudo-random test program generator”, IBM Systems Journal, Vol. 30, No. 4, 1991, pp. 527-538.

[16] W.G. Spruth, “The Design of a Microprocessor”, Springer Verlag 1989.

[17] A. Chandra. Et al., “AVPGEN - A Test Generator for Architecture Validation”, IEEE Transactions on VLSI Systems, June 1995, Vol. 3, No. 2, pp. 188-200.

[18] A. Aharon et al., “Test Program Generation for Functional Verification of PowerPC Processors in IBM”, Proc. of the 32nd Design Automation Conf. 1995, pp. 279-285.

[19] B.O’Krafka et al., “MPTG: A Portable Test Generator for Cache-Coherent Multiprocessors”, Intl. Phoenix Conf. On Comp. And Comm. 1995, pp. 38-44.

[20] A. Saha, “A Simulation-Based Approach to Architectural Verification of Multiprocessor Systems”, Intl. Phoenix Conf. On Comp. And Comm. 1995, pp. 34-37.

[21] J. Darringer, “The Application if Program Verification Techniques to Hardware Verification” 16th DAC, 1979, pp. 375-381.

[22] W.C. Carter, W. Joyner, D. Brand, “Microprogram Verification considered necessary”, National Computer Conference 1978, pp. 657-664.

[23] G. L. Smith, R. J. Bahnsen, H. Halliwell, “Boolean Comparison of Hardware and Flowcharts”, IBM J. Res. Develop., Vol. 26, January 1982, pp. 106-116.

[24] A. Kuehlman et al., “Verity - A formal verification program for custom CMOS circuits”, IBM J. Res. Develop., Vol. 39, No. 1/2, Jan/March 1995, pp. 149-165.

[25] K. L. MCmillan, “Symbolic Model Checking”, ISBN: 0-7923-9380-5, Kluwer, 1993

[26] I. Beer et al., “RuleBase: an Industry-oriented Formal Verification Tool”, Proc. of the 33rd Design Automation Conf. 1996, pp. 655-660.

[27] T. Schlipf et al., “Formal verification made easy”, IBM J. Res. Develop., Vol 41, No. 4/5, July/Sept. 1997, pp. 567-576.

[28] K. Diefendorf, ”Power4 Focuses on Memory Bandwidth”, Microprocessor Report, Vol. 13, Number 13, October 6, 1999.

[29] R. B. Hitchcock, G.I. Smith, D.D. Cheng," Timing Analysis of Computer Hardware", IBM Journal RD, v26, No 1, pp. 100 - 105, January 1982
[30] D. Tom, "Automatic Delay Adjustment for Static Timing Analysis", U.S. Patent 5,210,700, 11 May 1993.
[31] L. Stok, D.S. Kung, D. Brand, A.D. Drumm, A.J. Sullivan, L.N. Reddy, N. Hieter, D.J. Geiger, H.H. Chao, and P.J. Osler, "BooleDozer: Logic Synthesis for ASICs", IBM Journal of Research and Development, vol. 40, no. 4, pp. 407-430 (July 1996).
[32] R.P. Abato, A.D. Drumm, and D.J. Hathaway, L.P.P.P. van Ginneken, "Incremental Timing Analysis", U.S. patent 5,508,937, 16 April, 1996.
[33] J.W. Goetz and D.J. Hathaway, "Timing Analysis Using Slack Stealing", IBM Technical disclosure Bulletin, vol. 22, no. 10B, March 1991.
[34] D.J. Hathaway, J.P. Alvarez, and K.P. Belkhale, "Network Timing Analysis Method which Eliminates Timing Variations between Signals Traversing a Common Circuit Path", U.S. patent 5,636,372, 3 June 1997.
[35] M.Karnaugh, "The Map Method for Synthesis of Combinational Logic Circuits", Trans AIEE, Pt I, Vol 72, no 9, pp. 593-599 (1953).
[36] T. D. Friedman and S. C.Yang, "Methods Used in an Automatic Logic Design Generator (ALERT), "IEEE Trans Computers, C18, pp. 593-614 (1969).
[37] S.J. Hong, R.G. Cain, D.L. Ostapko, "MINI: A Heuristic Approach for Logic Minimization", IBM Journal of Research and Development, vol 18, No 5, pp. 443-458 (Sept 1974).
[38] R.K. Brayton, C.T. McMullen, "The Decomposition and Factorization of Boolean Expressions", Proc ISCAS April, 1982.
[39] J. A. Darringer, W. H. Joyner, "A New Look at Logic Synthesis",Proc 17th Design Automation Conf, June 1980, pp. 543-549.
[40] D. Brand: "Redundancy and Don't Cares in Logic Synthesis", IEEE Transactions on Computers, Vol C-32, No. 10, October 1983, pp.947-952.
[41] L.H. Trevillyan, W. H. Joyner, C. L. Berman, "Global Flow Analysis in Automatic Logic Design", IEEE Transactions on Computers, v C-35, pp. 77-81, January, 1986.
[42] S. Kundu, L. Huisman, I. Nair, V. Iyengar, L. Reddy, "A Small Test Generator for Large Designs", Proceedings of the IEEE International Test Conference, pp. 30-40 (September 1992).
[43] R. Puri, A. Bjorksten, and T. E. Rosser, "Logic Optimization by Output Phase Assignment in Dynamic Logic Synthesis", ACM/IEEE International Conference on CAD, 1996.
[44] D. Kung "A fast fan-out correction algorithm for near continuous buffer libraries", p. 353-355, DAC 98.
[45] L. Stok, M. A. Iyer,A.J. Sullivan, "Wavefront Technology Mapping" ,Proc of Design Automation and Test in Europe 1999 (March 1999).
[46] L.P.P.P. van Ginneken, "Buffer Placement in Distributed RC-tree Networks for Minimal Elmore Delay", 1990 International Symposium on Circuits and Systems, pp. 865-868.
[47] P. Kudva, W. Donath, L.M Reddy, L. Stok, P Villarubia, "Transformational Placement and Synthesis" Proc of Design Automation and Test in Europe 2000, March 2000
[48] R.Bergamaschi, R. O'Connor, L. Stok, M Moricz, S. Prahash, A. Kuehlmann, D. Rao, "High-level synthesis in an Industrial Environment", IBM Journal of Research and Development, Vol 39, no ½ January 1995.
[49] S. Raje and R. Bergamaschi, "Generalized Resource Sharing", Proceedings of the International Conference on Computer-Aided Design", pp. 326-332, November 1997.
[50] Franklin Branin, "Computer Methods of Network Analysis", Proceedings of IEEE, vol. 55, no. 11, November 1967, pp. 1787-1801.
[51] Gary D. Hachtel, Robert K. Brayton, and Fred Gustavson, "The Sparse Tableau Approach to Network Analysis and Design", IEEE Transactions on Circuit Theory, vol. CT-18, no. 1, January 1971, pp. 101-113.
[52] Chung-Wen Ho, Albert E. Ruehli, and Pierce A. Brennan, "The Modified Nodal Approach to Network Analysis", IEEE Transactions on Circuits and Systems, vol. CAS-22, no. 6, June 1975, pp. 504-509.
[52] IBM Advanced Statistical Analysis Program, ASTAP, general info. manual GH20-1271-0, IBM.
[54] E. Lelarasmee, Albert E. Ruehli, Alberto L. Sangiovanni-Vincentelli, "The Waveform Relaxation Method for the Time-domain Analysis of Large Scale Integrated Circuits", IEEE Trans. CAD of Integrated Circuits and Systems., CAD-1, 1982, pp. 131-145.
[55] PowerSPICE User's Guide, manual PSP3-1000-02, July 1999.
[56] B.R. Chawla, H.K.Gummel, aand P. Kozak, "MOTIS: An MOS timing simulator, "IEEE Trans. on Circuits and Systems, Dec. 1975, pp. 901-910.

[57] C. Visweswariah and R. A. Rohrer, “Piecewise Approximate Circuit Simulation,” IEEE Trans. on Computer Aided Design, July 1981, pp. 861-870.
[58] A. Devgan and R. A. Rohrer, “Adaptively Controlled Explicit Simulation, “IEEE Trans. on Computer Aided Design, June 1994, pp. 746-762.
[59] A.E.Ruehli, “ Inductance calculations in a complex integrated circuit environment,” IBM J. Res. Develop., Vol. 16, No. 5, Sept. 1972, pp. 470-481.
[60] A.E. Ruehli and P.A. Brennan, “ Efficient capacitance calculations for three-dimensional multiconductor systems,” IEEE Trans. Microwave Theory Tech., Vol. MTT-21, Feb. 1973, pp. 76-82.
[61] A.E. Ruehli, “ Equivalent circuit models for three-dimensional multiconductor systems,” IEEE Trans. Microwave Theory Tech., Vol MTT-22, pp. 216-221, Mar. 1974.
[62] P. Brennan, et al., “ Three-dimensional inductance computations with partial element equivalent circuits,” IBM J. Res. Develop., Vol 23, pp. 661-668, Nov. 1979.
[63] W.T. Weeks, “ Calculation of coefficients of capacitance of multiconductor transmission lines in the presence of a dielectric interface,” IEEE Trans. Microwave Theory Tech., vol. MTT-18, Jan. 1970, pp. 35-43.
[64] W.T. Weeks, et al., “ Resistive and inductive skin effect in rectangular conductors,” IBM J. Res. Devlop., Vol. 23, pp. 652-660, Nov. 1979.
[65] COSMIC user guide, Manual 0220-5535-00, December 1993.
[66] K.L. Shepard, V. Narayanan, and R. Rose, “Harmony: Static analysis of deep sub-micron digital integrated circuits”, IEEE Transactions on CAD, vol. 18, no 8., pp. 1132-1150, Aug. 1999.
[67] B. Krauter and S. Mehrotra, “Layout based frequency dependent inductance and resistance extraction for on-chip interconnect timing analysis”, Proc. of Design Automation Conf., p. 303-308, 1998.
[68] Howard Chen, “Interconnect and circuit modeling techniques for full chip power supply noise analysis”, IEEE Trans. Component, Package, Manufacturing Technology. B, Adv. Package., vol.21, no.3, pp. 209-15, Aug. 1998.
[69] C. Visweswariah, “Optimization techniques for high-performance digital circuits”, Proceedings of IEEE International Conference on Computer Aided Design (ICCAD), San Jose, CA, USA, pp., 198-207, Nov. 1997.
[70] A. R. Conn, I. M. Elfadel and W. W. Molzen, Jr., P. R. O’Brien, P. N. Strenski, C. Visweswariah and C. B. Whan, “Gradient-based optimization of custom circuits using a static-timing formulation”, Proc. of the Design Automation Conference”, p. 452-459, 1999.
[71] C. Visweswariah and A. R. Conn, “Formulation of static circuit optimization with reduced size, degeneracy and redundancy by timing graph manipulation”, Proc. of the Int. Conf. on Computer Aided Design, 1999, p. 244-251, 1999.
[72] G. Katopis, D. Becker and H. Stoller, “First level package design considerations for the IBM’s S/390 G5 Server”, Tech. Digest, IEEE Topical Meeting on Electrical Performance of Electronic Packaging, Oct. 1998, pp. 15-16.
[73] D. L. Rude, “Statistical method of noise estimation in a synchronous system”, IEEE Transactions on Components, Packaging, and Manufacturing Technology Part B: Advanced Packaging Vol. 17, No. 4, Nov 1994, pp 514-519.
[74] K.H. Khokhani, A.M. Patel,”The Chip layout problem: A placement procedure for LSI”, in Proc. 14th Annual Design Automation Conf. (1977), pp.291-297
[75] A.W. Aldridge, R.F. Keil, J.H. Panner, G.D. Pittman, and D.R. Thomas, “A 40K Equivalent Gate CMOS Standard-Cell Chip”, IEEE 1987 Custom Integrated Circuit Conference, pp. 248-252.
[76] J.H. Panner, R.P. Abato, R.W. Bassett, K.M. Carrig, P.S. Gillis, D.J. Hathaway, and T. W. Sehr, “A Comprehensive CAD System for High-performance 300K-Circuit ASIC Logic Chips”, IEEE Journal of Solid-State Circuits, vol. 26, no. 3, pp. 300-309, March 1991.
[77] J.Y. Sayah, et.al. “Design planning for high-performance ASICs”, IBM Journal of Research and Development, Volume 40, Number 4, pp 431-452, July 1996
[78] A. Hetzel : A Sequential Detailed Router for Huge Grid Graphs. Design, Automation and Test in Europe, Proceedings, IEEE 1996, 332-338
[79] C. Albrecht, B. Korte, J. Schietke, J. Vygen : Cycle Time and Slack Optimization for VLSI-Chips. Proceedings of the IEEE International Conference on Computer-Aided Design (1999), 232-238
[80] S. Kirkpatrick, C.D. Gelatt, Jr., M.P. Vecchi, ”Optimization by Simulated Annealing”, Science, Volume 220, Number 4598. May 13, 1983

[81] J. Vygen : Algorithms for Large-Scale Flat Placement. Proceedings of the 34th Design Automation Conference, ACM 1997, 746-751
[82] J. Vygen : Algorithms for Detailed Placement of Standard Cells. Design, Automation and Test in Europe, Proceedings, IEEE 1998, 321-324
[83] R.S. Belanger, D.P. Conrady, R.E. DuBois, W.R. Heller, P.S. Honsinger, T.J. Lavery, G.W. Mahoney, G.F. Miceli, S.J. Rothman, E.C. Schanzenbach, C.R. Selinger, D. Sitaram, "Enhanced Chip/Package Design for the IDM ES/390",ICCD.
[84] E. E. Davidson, "Electrical Design of a High Speed Computer Package," IBM J. of Res. & Dev., Vol. 26 No. 3, pp. 349-361, May 1982.
[85] K.A. Chen, M. Feuer, K.H. Khokhani, N. Nan, S. Schmidt,"The chip layout problem: An automatic wiring procedure", in Proc. 14th Annual Design Automation Conf. (1977), pp. 298-302
[86] W.R. Heller, W.F. Mikhail, W.E. Donath, "Prediction of Wiring Space Requirements for LSI", Proc. 14th Annual Design Automation Conf. (1977), pp.32-42
[87] W.E. Donath, Wire length distribution for placements of computer logic, IBM Journal of Research and Development Volume 25, Number 3, May 1981
[88] R.R. Habra, "Interactive Graphics for Wiring", Proceedings of the International Conference on Interactive Techniques in Computer Aided Design (1978), pp.317-320
[89] H.H. Chen, J.S. Neely, "Interconnect and circuit modeling techniques for full-chip power supply noise analysis," IEEE Transactions on Components, Packaging, and Manufacturing Technology - Part B: Advanced Packaging, vol. 21, no. 3, pp. 209-215, August 1998.
[90] K.M. Carrig, D.J. Hathaway, K.W. Lallier, J.H. Panner, and T.W. Sehr, "Method and Apparatus for Making a Skew-Controlled Signal Distribution Network", U.S. Patent 5,339,253, 16 August, 1994.
[91] "R.-S. Tsay, "An Exact Zero Skew Clock Routing Algorithm", IEEE Trans. CAD, vol. 14, no. 12, pp. 242-249, February 1993.
[92] D.J. Hathaway, R.R. Habra, E.C. Schanzenbach, and S.J. Rothman, "Circuit Placement, Chip Optimization, and Wire Routing for IBM IC Technology", IM Journal of Research and Development, vol. 40, no. 4, pp. 453-460, 4 July 1996.
[93] P.J. Restle, A. Deutsch, "Designing the Best Clock Distribution Network", 1998 Symposium on VLSI Circuits, pp. 2-5.
[94] W.E. Donath and J. Lesser, "LAGER, A language for the Digital Transcription of Design Patterns, " Research Report RC1730, IBM T. J. Watson Research Center, Yorktown Heights, NY 1966.
[95] J.S. Koford, G.A. Sporzinsky and P. R. Strickland, "Using a Graphic Data Processing System to Design Artwork for Manufacturing Hybrid Integrated Circuits", Proceedings of the Fall Joint Computor Conference, San Francisco CA, 1966, pp.229-246
[96] P. Carmody, A Barone, J. Morrell, A. Weiner and J. Hennesy, " An Interactive Graphics System for Custom Design", Proceedings of the 17th Design Austomation Conference, Minneapolis, MN, 1980 pp. 296-308.
[97] C. McCaw "Unified Shapes Checker - A Checking Tool for LSI", Proceedings of the 16th Annual Design Automation Conference, San Diego CA 1979, pp. 81-87
[98] D. Lambert "Graphics Language/One-IBM corporate-wide physical design data format". ACM/IEEE 18th Design Automation Conference Proceedings. Nashville, TN, June 1981. pp. 713-719.
[99] L. Liebmann, B. Grenon, M. Lavin, T. Zell. "Optical proximity correction: a first look at manufacturability". Microlithography Woeld, Vol. 4, No. 2, pp 7-11, 1995.
[100] M. A. Lavin, W. C. Leipold, "VLSI Manufacturing Shape Data Preparation", MicroNews, Vol. 5, No. 3, pp.14-17, 1999.
[101] F. Heng, Z. Chen, G. Tellez, "A VLSI Artwork Legalization Technique Based on a New Criterion of Minimum Layout Perturbation", ISPD-97, pp.116-121.
[102] C.H. Stapper, "Modeling of Defects in Integrated Circuit Photolithographic Patterns", IBM Journal of Research and Development, Vol. 28, No. 4, July 1984, pp. 461-475.
[103] C.H. Stapper, A.N. McLaren, and M. Dreckman, "Yield Model for Productivity Optimization of VLSI Memory Chips with Redundancy and Partially Good Product", IBM Journal of Research and Development, Vol. 24, No. 3, May 1980, pp. 398-409.
[104] G. S. Ditlow, D. R. Dooling, D. E. Moran, R. L. Moore, G. E. Tellez, R. J. Williams, T. W. Wilkins, "Method to Improve Integrated Circuit Defect Limited Yield", Patent Pending BU9-98-140, 1998.

[105] G. E. Tellez, A D. Dziedzic, A. J. Allen, D. N. Maynard, W. E. Donath, D. M. Newns, M. A. Lavin, “A Method for Prediction of Random Defect Yields of Integrated Circuits with Accuracy and Computation Time Controls”. Patent Pending BU9-99-191, 1999.
[106] G. E. Tellez, G. R. Doyle, R. B. Wilcox Jr, G. Starkey, P. S. Honsinger, C. L. Meiley, S. G. Lovejoy, S.G. “A Method for Improving Wiring Related Yield and Capacitance Properties of Integrated Circuits by Maze-Routing”, Patent Pending FIS9-1999-0154, 1999
[107] N. Rohrer, C. Akrout, M. Canada, D. Cawthron, B. Davari, R. Floyd, S. Geissler, R. Goldblatt, R. Houle, P. Kartschoke, D. Kramer, P. McCormick, G. Salem, R. Schulz, L. Su, L. Whitney, “A 480MHz RISC Microprocessor in a 0.12um Leff CMOS Technology with Copper Interconnects”, 1998 ISSCC Digest of Technical Papers, Feb. 1998, pp. 240-241.
[108] K. Maling, and E.L. Allen, “A Computer Organization and Programming System for Automated Maintenance”, IEEE Transactions on Electronic Computers, Vol. EC12, No. 6, December 1963, pp. 887-895.
[109] E.B. Eichelberger, “Method of Level-Sensitive Testing a Functional Logic System”, U.S. Patent No. 3,761,695, September 25, 1973.
[110] E.B. Eichelberger, and T.W. Williams, “A Logic Design Structure for LSI Testability”, Proceedings, 14th Design Automation Conference, 1977, pp. 462-468.
[111] J. Reilly, A. Sutton, R. Nasser, and R. Griscom, “Processor Controller for the IBM 3081” IBM Journal of Research and Development, Vol. 26, No. 1, January 1982, pp. 22-29.
[112] P. Goel, “An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits”, IEEE Transactions on Computers, Vol. C-30, March 1981, pp. 215-222.
[113] J.P. Roth, “Diagnosis of Automata Failures: A Calculus and a Method”, IBM Journal of Research and Development, Vol. 10, July 1966, pp. 278-291.
[114] P.S. Bottorf, H. Godoy, and G.B. Franklin, “Automatic Checking of Logic Design Structures for Compliance with Ground Rules”, Proceedings, 14th Design Automation Conference, June 1977, pp. 469-478.
[115] P.H. Bardell, and W.H. McAnney, “Self-Testing Multichip Modules”, Proceedings, IEEE International Test Conference, 1982, pp. 200-204.
[116] Z. Barzilai, J.L. Carter, B.K. Rosen, and J.D. Rutledge, “HSS - High Speed Simulator”, IEEE Transactions on Computer-Aided-Design, Vol. CAD-6, July 1987, pp. 601-617.
[117] B.L. Keller, and T.J. Snether, “Built-In Self-Test Support in the IBM Engineering Design System”, IBM Journal of Research and Development, Vol. 34, No. 2/3, March/May 1990, pp. 406-415.
[118] J.A. Waicukauski, E.B. Eichelberger, D.O. Forlenza, E. Lindbloom, and T. McCarthy, “Fault Simulatin for Structured VLSI”, VLSI Systems Design, December 1985, pp. 20-35
[119] F. Motika, and J. Waicukauski, “Weighted Random Pattern Testing Apparatus and Method”, U.S. Patent No. 4,688,233, August 18, 1987.
[120] E.B. Eichelberger, “Method of Propagation Delay Testing a Functional Logic System”, U.S. Patent No. 3,784,907, January 8, 1974.
[121] E.P. Hsieh, R.A., Rasmussen, L.J. Vidunas, and W.T. Davis, “Delay Test Generation”, Proceedings, 14th Design Automation Conference, June 1977, pp. 486-491.
[122] F. Motika et al., “ A Logic Chip Delay-Test Method Based on System Timing”, IBM Journal of Research and Development, Vol. 34, No. 3/4 1990, pp.299-324.
[123] B. Koenemann, J. Barlow, P. Chang, R. Gabrielson, C. Goertz, B. Keller, K. McCauley, J. Tisher, V. Iyengar, B. Rosen, and T. Williams, “Delay Test: The Next Frontier for LSSD Test Systems”, Proceedings, IEEE International Test Conference, 1992, pp. 578-587.
[124] J.A. Waicukauski, E. Lindbloom, B.K. Rosen, and V.S. Iyengar, “Transition Fault Simulation”, IEEE Design and Test of Computers, 1987, pp 32-38.
[125] Y. Aizenbud, P. Chang, B. Koenemann, V. Iyengar, M. Leibowitz, M., D. Smith, and B. Rosen, “AC Test Quality: Beyond Transition Fault Coverage”, Proceedings, International Test Conference, 1992, pp. 568-577.
[126] G.L. Smith, “Model for Delay Faults Based on Paths”, Proceedings, IEEE International Test Conference, 1985, pp. 342-349.
[127] E.B. Eichelberger, E.I. Muehldorf, T.W. Williams, R.G. Walther, “A Logic Design Structure for Testing Internal Arrays”, 3rd USA-JAPAN Computer Conference, 1978, pp. 266-272.
[128] E.B. Eichelberger, “Testing Embedded Arrays”, U.S. Patent No. 3,961,252, June 1, 1976.
[129] D. Westcott, “ The Self-Assist Test Approach to Embedded Arrays”, IEEE International Test Conference, 1981, pp. 203-207.

[130] P. Goel, and M.T. McMahon, “Electronic Chip-In-Place Test”, IEEE International Test Conference, 1982, pp. 83-90.
[131] J. LeBlanc, “LOCST: A Built-In Self-Test Technique”, IEEE Design and Test, November 1984, pp. 45-52.

[132] P. Gillis, F. Woytowich, K. McCauley, and U. Baur, “Delay Test of Chip I/Os Using LSSD Boundary Scan”, IEEE International Test Conference, 1998, pp. 83-90.
[133] R.W. Bassett, B.J. Butkus, S.L. Dingle, M.R. Faucher, P.S. Gillis, J.H. Panner, J.G. Petrovick, and D.L. Wheater, “Low Cost Testing of High-Density Logic Components”, Proceedings, IEEE International Test Conference, 1989, pp. 550-557.
[134] O.A. Torreiter, U. Baur, G. Goecke, and Kevin Melocco, “Testing the Enterprise IBM System/390 Multi-Processor”, IEEE International Test Conference, 1997, pp. 115-123.
[135] B.L. Keller, “Hierarchical Pattern Faults for Describing Logic Circuit Failure Mechanisms”, U.S. Patent No. 5,546,408, August 1996.
[136] P.Gillis, T. Guzowski, B. Keller, and R. Kerr, “Test Methodologies and Automation for IBM ASICs”, IBM Journal of Research and Development, Vol. 40, No. 4, July 1996, pp. 461-474.
[137] S. Kundu, A. Kuehlmann, and A. Srinivasan, “CMOS transistor network to gate level model extractor for simulation, verification and test generation”, U.S. Patent No. 5,629,858, May 1997.
[138] W. Huott, T. Koprowski, B. Robbins, M. Kusko, S. Pateras, D. Hoffman, T. MacNamara, and T. Snethen, “Advanced Microprocessor Test Strategy and Methodology”, IBM Journal of Research and Development, Vol. 41, No. 4/5, July/September 1997, pp. 611-627.
[139] Y. Arzoumanian, and J.A. Waicukauski, “Fault Diagnosis in an LSSD Environment”, Digest of Papers, International Test Conference, 1981, pp. 362-370.
[140] J.A. Waicukauski, V.P. Gupta, and S.T. Patel, “Diagnosis of BIST Failures by PPSFP Simulation”, Proceedings, IEEE International Test Conference, 1987, pp. 480-484.
[141] T.W. Williams, and N.C. Brown, “Defect Level as a Function of Fault Coverage”, IEEE Transactions of Computers, Col. C-30, December 1981, pp.987-988.
[142] D.S. Cleverly, “Product Quality Level Monitoring and Control for Logic Chips and Modules”, IBM Journal of Research and Development, Vol. 27, No. 1, January 1983, pp. 4-10.
[143] C.H. Stapper, A.N. McLaren, and M. Dreckman, “Yield Model for Productivity Optimization of VLSI Memory Chips with Redundancy and Partially Good Product”, IBM Journal of Research and Development, Vol. 24, No. 3, May 1980, pp. 398-409.
[144] E.K. Vida-Torku, W. Reohr, J.A. Mozel, and P. Nigh, “ Bipolar, CMOS, and BiCMOS Circuit Technologies Examined for Testability”, Proceedings of the 34th Midwest Symposium on Circuits and Systems, Vol. 2, 1992, pp. 1015 -1020.
[145] C.H. Stapper, “Modeling of Defects in Integrated Circuit Photolithographic Patterns”, IBM Journal of Research and Development, Vol. 28, No. 4, July 1984, pp. 461-475.
[146] P. Nigh, D. Vallett, A. Patel, J. Wright, F. Motika, D. Forlenza, R. Kurtulik, and W. Chong, “Failure Analysis of Timing and IDDq-only Failures from the SEMATECH Test Methods Experiment”, International Test Conference, 1998, pp. 43-52.
[147] A.H. Farrahi, D.J. Hathaway, M. Wang, M. Saffrazadeh, “Quality of EDA CAD Tools: Definitions, Metrics, and Directions”, Invited paper, 2000 IEEE International Symposium on Quality of Electronic Design, March, 2000.
[148] D.J. Hathaway, “Timing Analysis Method for PLLS”, U.S. patent 5,944,834, issued 8/31/1999.
[149] D. Garrett, M. Stan, and A. Dean, “Challenges in Clock gating for a Low Power ASIC Methodology”, 1999 IEEE/ACM International Symposium on Low Power Electronics and Design, pp. 176-181.
[150] W. Huott, T. Koprowski, B. Robbins, M. Kusko, S. Pateras, D. Hoffman, T. MacNamara, and T. Snethen, “Advanced Microprocessor Test Strategy and Methodology”, IBM Journal of Research and Development, Vol. 41, No. 4/5, July/September 1997, pp. 611-627.
[151] W. Huott, T.J. Slegel, T. Lo, and P. Patel, “Programmable Computer System Element with Built-In Self Test Method and Apparatus for Repair During Power-On”, U.S. Patent No. 5,659,551, May 1996.