Standard-cell-based design methodology for high-performance support chips

by B. Kick
U. Baur
J. Koehl
T. Ludwig
T. Pflueger

We describe the methodology used for the design of a set of CMOS support chips used in the IBM S/390® Parallel Enterprise Server Generations 3 and 4. The logic design is based on functional units, and the majority of the logic is implemented by standard cell elements placed and routed flat, using timing-driven techniques. Custom library elements are used wherever needed for performance reasons. Using this approach, a density has been achieved that is comparable to those of contemporary custom designs, combined with very attractive turnaround times.

Introduction

Custom design is the dominant design style for high-performance processors. It offers the advantage of full control over the size and the location of each transistor for performance tuning, but requires considerable effort to implement because of the complexity of a complete transistor-level design. This complexity creates the need to introduce additional hierarchies, usually leading to a "floorplanning" approach.

A standard cell design approach (**Figure 1**) makes it possible to globally apply advanced optimization algorithms, which reduce the manual effort required and improve the quality of the synthesized logic during layout. The use of basic standard cell elements reduces complexity to the extent that a complete chip design can be handled flat by layout and test generation tools, removing the need for artificial floorplan boundaries. Our approach uses a small number of custom logic macros and custom memory arrays whenever a standard cell solution is not competitive. The major part of the combinational logic portions, however, are implemented in standard cells.

Design entry, synthesis, and simulation are performed on the basis of functional units. There is no need to optimize logic partitioning on the basis of timing, layout, and test considerations. Flat, timing-driven placement and routing without floorplan boundaries minimizes interconnection delay in critical paths. This, coupled with in-place logic optimization, achieves a post-layout cycle time no more than 15% above the zero-net estimate.

The testing methodology we have used consists of design for test (DFT) to ensure high test coverage, and test pattern generation to enable testing, analysis, and debugging of chips in manufacturing. Key are fast turnaround time and high-quality testing.

Test data generation, circuit and logic design, and timing verification are performed with proprietary IBM tools [1–4]. The tools for layout optimization were

0018-8646/97/$5.00 © 1997 IBM

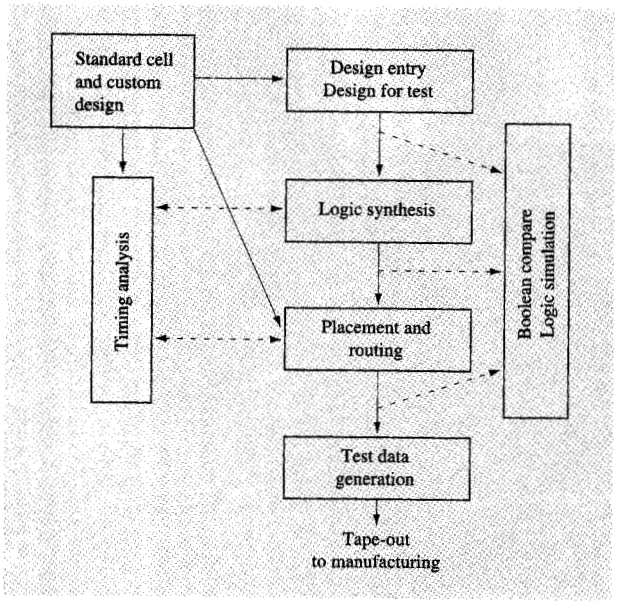

Figure 1

Overview of design flow.

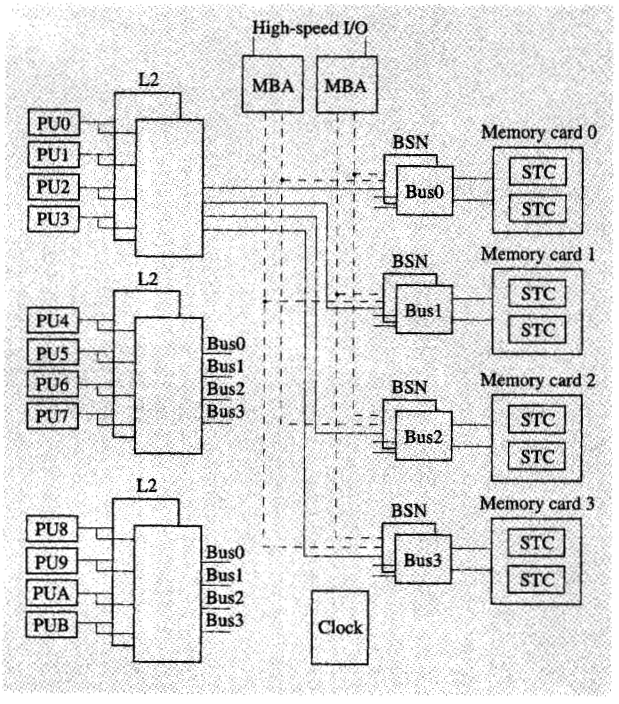

Figure 2

System overview.

developed at the Institute for Discrete Mathematics at Bonn, Germany, in close cooperation with the IBM Laboratory in Boeblingen, Germany. This cooperation has minimized the lead time required to incorporate combinatorial optimization research results into our production tools. The approach has been used successfully on a set of CMOS chips which, together with processor and cache, are the heart of the S/390* Parallel Enterprise Server Generations 3 and 4.

System overview

The chip set (**Figure 2**) consists of processor chips (PU0–PUB), cache chips (L2), and a set of support chips (Clock, MBA, BSN, STC). A tightly coupled S/390 multiprocessor system with up to twelve processors and 16GB physical main memory can be designed with this chip set. The clock chip (Clock) provides the clocks, self-test, and power-on control logic, and the interface with the service element for all chips in the system. The memory bus adapter (MBA) chips are direct-memory-access (DMA) controllers that are the interface between the asynchronous, byte-serial I/O buses and the 16-byte-wide system bus. The bus-switching network (BSN) chips hold shared level-3 caches and bus arbiters that control the concurrent access of PUs, MBAs, and system-wide memory. The storage controller (STC) chips are DRAM controllers, supporting transparent refresh, interleaving, and multibit error detection and repair. More details can be found in [5].

Technology and design of custom elements

- *Technology*

The CMOS process [6, 7] used on the chip set was developed by the IBM Microelectronics Division. The technology provides six layers of metallization—one layer for internal circuit wiring only, and four layers for wiring in a 1.8-μm wiring pitch. The last metallization layer is used primarily for wiring redistribution to the chip I/O pads. The technology parameters are shown in **Table 1**.

- *Library and chip image*

The standard cell library we used provides a set of logic gates, latches, and I/O cells which fit into 3.5 million placement locations and are interconnected through horizontal and vertical wiring tracks defined by the chip image. The I/O cells can be placed anywhere among the 3.5 million legal locations. After chip placement and routing, the unused cell locations are filled with nonpersonalized gate array elements to provide an engineering change capability with metallization changes only.

• *Custom circuit design*

The base standard cell library provides simple logic gates, but a small set of custom logic macros and custom SRAM macros was required for the special needs of the S/390 in order to improve cycle time and density. The custom implementation of the macros gives the circuit designer the freedom to use special circuit design techniques such as dynamic and double-pass circuits [8] to improve the propagation delay.

The circuit design flow (**Figure 3**) begins with a specification sheet defining the macro requirements. With this information, a model written in a proprietary hardware description language (HDL) is designed [9]. This HDL model defines the logic behavior and must be as compact as possible to reduce logic simulation time. The HDL model is thoroughly simulated against the specification sheet and becomes the "golden" model for the following design process. All other design sources required on the way to layout are checked against the golden model.

The first step of the schematic-driven layout is the implementation of the logic function in transistors with a schematic entry tool [10]. An iterative process based on transistor-level simulation followed by transistor modifications is necessary to meet the timing, performance, and power-consumption targets of the macro.

A Boolean equivalence checker [11] compares the transistor schematics against the golden model and gives early simulation-independent feedback of the correct implementation. An early timing model is generated for the chip-level delay calculator [4]. This early timing model is replaced later in the design process by the final timing model, based on information extracted from the circuit's layout. The device and net information in the schematics is used by a proprietary schematic-driven layout tool. Compliance of the macro layout with the technology design rules is checked with a hierarchical design rule check (DRC). The layout design style could vary, from a full shape-by-shape design to the use of circuit generators for base logic functions such as NANDs.

To support generation of the chip place and route rules, additional shape and text information must be added to the layout design. After this process, the custom macros can be used like big standard cell circuits, placeable in any legal location. Providing signal-pin, power-pin, and blockage information to the physical design tools allows automatic power and signal wiring at the chip level.

The custom macro layout is fed into a proprietary layout parasitic extraction (LPE) tool. The transistor geometries (width and length), as well as all parasitic elements such as diffusion capacitances and line-to-line capacitances, are then extracted from the layout. The generated netlist with parasitic elements is used for transistor-level resimulation to ensure that the function and performance are still correct. This netlist is the source for the final, most accurate timing model of the macro.

After the custom macro layout is complete, a final layout versus schematic (LVS) check is performed. This check generates a layout netlist and compares it against the schematic netlist, not only checking network topology and device sizes, but also detecting net opens and shorts.

Finally a test model is generated, breaking down all transistor schematics into the primitive functions understood by test pattern generation (TPG), such as AND, NAND, NOR, OR, and XOR. This model is verified against the golden HDL model to guarantee logic equivalence between the implementations.

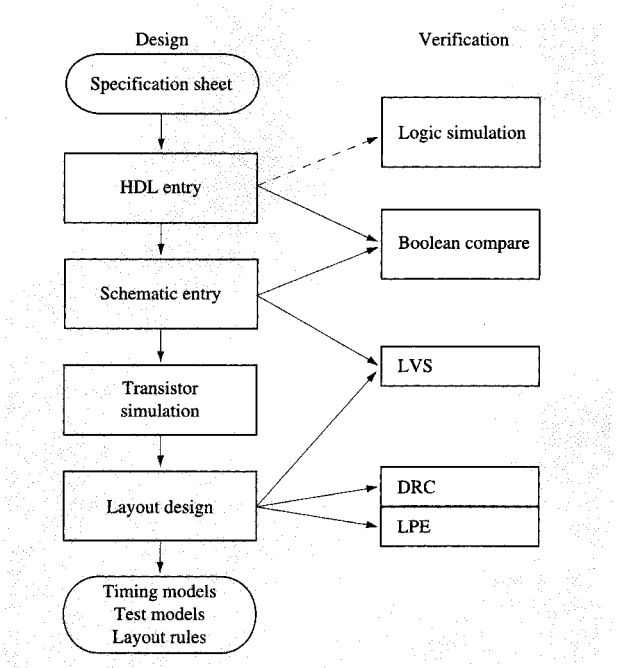

Figure 3

Circuit design flow.

Table 1 Technology parameters.

Feature	*Value*
Supply voltage	2.5 V
L_{eff}	0.25 μm
Minimum feature size	0.33 μm
T_{ox}	7 nm
Metal layers	1 + 5

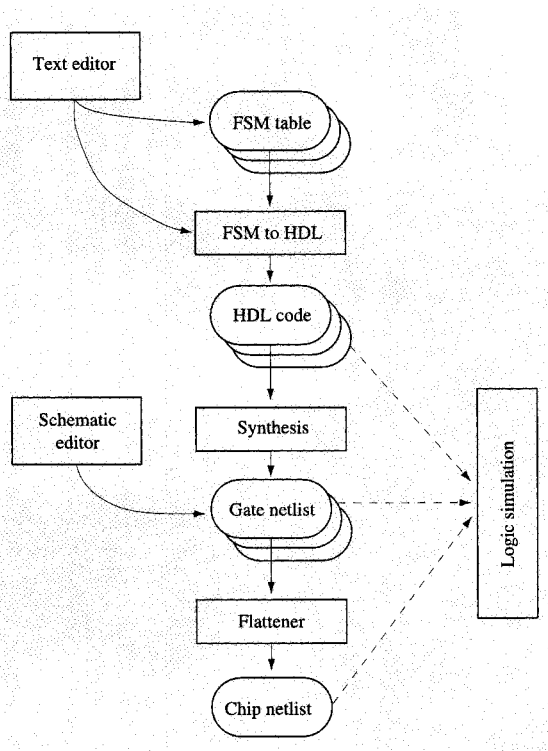

Figure 4

Logic design flow.

Design entry, synthesis, and simulation

• *Design entry*

The design system accepts design data in three forms: gate-level schematics, hardware design language (HDL) code, and finite-state machine (FSM) tables (**Figure 4**).

Gate-level schematics are preferred for data-flow-dominated designs and for designs that require careful manual design and optimization. Most parts of the processor and the L2 cache chip are designed at the gate level. The schematics are entered using a proprietary schematic editor that translates the schematics into gate-level netlists. Apart from macro expansion, this is a one-to-one translation; no logic optimization is performed.

HDL code and FSM tables are preferred for control-flow-dominated designs. Most parts of the support chips are HDL code or FSM table designs. HDL code is a proprietary hardware-description language [9]. The level of description is similar to the concurrent subset of VHDL: Boolean expressions, signal assignments, component instantiations, etc.

FSM tables are convenient because they describe finite-state machines more compactly than HDL code. FSM tables are translated to HDL code for synthesis and simulation. For simulation, the generated HDL code is instrumented to collect statistics about state transitions exercised by a set of test cases. This information is used to create test cases that exercise all possible state transitions.

• *Logic synthesis*

The logic synthesis system, BooleDozer*, reads the HDL code and generates gate-level netlists. BooleDozer performs technology-independent optimization, technology mapping, and timing optimization to generate a netlist of minimal size that meets the delay objectives [2, 3]. Synthesis uses the same delay calculator as placement and routing, with the exception that interconnection capacitances and resistances are estimated as a function of fanout, based on statistics from placement and routing.

Because a full-chip design cannot be synthesized in one run, it must be partitioned into pieces of a few thousand synthesizable gates each. This approach has the advantage that synthesis jobs can run in parallel on multiple machines, reducing turnaround times. Typically, synthesis times range from one to ten hours of CPU time per partition, resulting in overnight turnaround.

Partitioning requires that delay objectives for the chip be broken down into delay objectives for each partition. This process, designated as slack apportionment, assigns delay objectives to partitions in such a way that if each partition meets its delay objective, the chip also meets the delay objective. The process first runs on an unoptimized design to generate initial delay objectives. The design is then resynthesized and optimized with respect to initial delay objectives, and is fed into slack apportionment again to generate improved delay objectives. This is an expensive process because it requires multiple full-chip synthesis runs, but in practice after two or three iterations the delay objectives become stable. Experiments show that slack apportionment need only be rerun after major design changes, which do not occur very often. Logic synthesis and schematic entry generate one netlist for each chip partition. The partition netlists are finally flattened into one chip netlist for flat placement and routing.

• *Simulation*

Extensive logic simulation at the unit, chip, and system level is performed to verify the functional correctness of the designs [12, 13]. Cycle-based simulation assumes zero delay, leaving timing verification to the delay calculator [4]. This approach nicely separates timing aspects from functional aspects and speeds up simulation considerably.

Unit-level and chip-level simulation are carried out using mostly HDL code models. This interactive mode of simulation is used primarily in the early stages of logic design to correct small design errors that are easy to detect. The bulk of simulation occurs at the system level.

System simulation uses gate-level models for the processor, cache, and memory interface chips, and behavior-level models for the I/O chips. A simulation monitor initializes the storage elements (latches, memories) of the model, loads test cases into the model, and provides tracing, assertion checking, and reporting capabilities. The monitor has a full-screen interface for interactive simulation, but most of the system simulation is done in batch mode. The simulation is performed in parallel with logic design, as soon as an initial, unit-level simulated design is available.

Chip placement and routing

• *Flat and timing-driven layout*

Because our chip-placement and routing tools have been refined over the course of four processor generations, and because of the tight cycle-time bounds imposed on the designs, the primary layout optimization objective has shifted from pure routability to cycle-time reduction.

To be able to judge the quality of a given layout, we needed a reasonable lower bound for the possible cycle time. A natural lower bound could be obtained by a static timing analysis of the logic network assuming a net length of zero for each net. In other words, each circuit drives the input capacitances of the next stage with interconnection length set to zero. Upon comparing the actual post-layout cycle time to this hypothetical zero-net cycle time using different design approaches such as floorplanning vs. flat, and timing-driven vs. connectivity-driven, we found that the approach that consistently produced the lowest interconnection delay was flat, timing-driven layout (**Figure 5**).

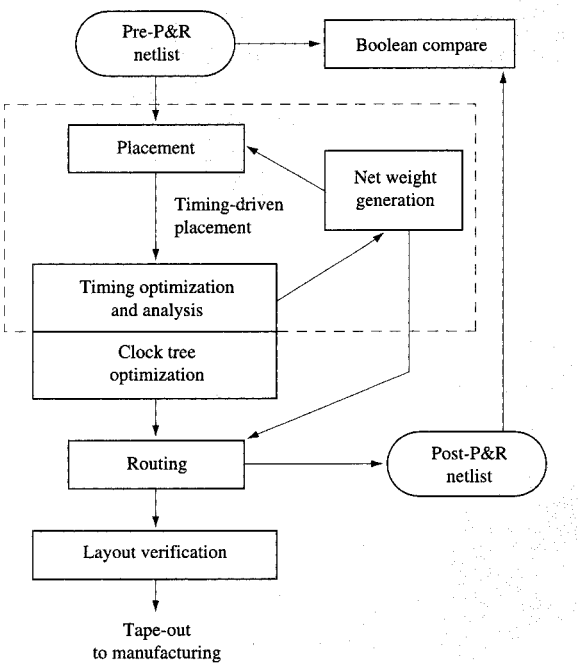

Figure 5

Layout flow.

• *Placement*

The ability to place and route complex designs flat and timing-driven is an important prerequisite for the design methodology presented here. This is made possible by quadratic optimization combined with a new quadrisection approach [14]. The approach computes net weights, derived from a concurrent timing analysis run, which are then used for the next optimization step. A description of detailed placement can be found in [15].

• *In-place optimization*

Logic optimization based on the actual placement is performed to further improve the cycle time. This is carried out in three steps:

1. *Clock synthesis* The clock tree is not considered during placement but is instead resynthesized after placement using a zero-skew approach similar to [16, 17]. Routing information for balanced routing is created as an input to the routing step.
2. *Power-level optimization* This is performed for timing optimization and power reduction. It uses one of the five power levels available for each standard-cell circuit.
3. *Buffer insertion* This is performed on timing-critical paths that still exceed the cycle-time limit after power-level optimization.

The resulting decisions are always based on actual placement data, as each circuit added is assigned to a placement location. Details on timing analysis and optimization techniques that are used can be found in [18].

• *Routing*

Special nets such as power buses and nets connected to I/O pads are routed first, and then congestion-driven global routing defines guide boxes for the following local routing step. The information generated during clock optimization drives the balanced routing of the clock nets. The ability to route the entire design flat removes the suboptimality introduced by the necessary pin propagation in a hierarchical approach. The routing tool supports different wire widths and separations and has a crosstalk analysis as well as removal capability [19, 20].

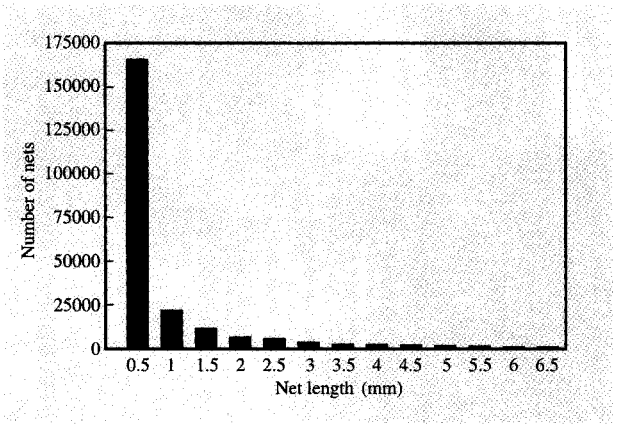

Figure 6

Net length distribution for the MBA chip.

Table 2 Chip statistics.

	BSN	*MBA*
Chip size (mm^2)	213	240
Transistors	16.6×10^6	3.6×10^6
Density (kTX/mm^2)	77.9	15.0
Standard cells	71×10^3	206×10^3
Custom macros	341	89
Pins	260×10^3	771×10^3
Nets	82×10^3	226×10^3
Total net length (m)	60.6	122
I/O pins	758	770
Cycle time (ns)	5.9	5.9

Table 3 Run times and memory for the MBA chip.

Layout step	*Run times on RS/6000* Model 590* (h)	*Memory* (MB)
Placement (2×)	16 (2×)	700
In-place optimization (2×)	25 (2×)	1200
Routing	26	300

- *Boolean compare and engineering changes*

To avoid any risk of introducing logic errors during in-place optimization, a Boolean equivalence checking tool [11] is used to verify the equivalence of the pre- and post-layout netlists. The design system supports late metallization-only changes by rerouting or by using gate-array circuits. This process is complicated by the fact that in-place optimizations during layout, and late functional changes by the logic designers, are carried out concurrently. We have implemented a flow to incorporate the functional changes performed on the pre-layout netlist into the post-layout netlist.

- *Results*

Figure 6 shows the net length distribution for the 15.5 × 15.5-mm^2 MBA chip. About 70% of the nets are less than 0.5 mm long, and very few nets are more than 5 mm in length. Restricting the functional units to floorplan regions would introduce global nets, which are typically longer. This relatively small increase in interconnection delay is an inherent advantage of our flat, timing-driven layout approach. On the most timing-critical paths we have been able to keep the ratio of post-layout to zero-net cycle time below 1.15. The actual ratio depends on the given technology, of course. It should also be noted that this ratio applies to only the most timing-critical paths. For less critical paths the contribution of interconnection delays is typically much larger, supporting the common industry view that for today's technologies, interconnection delay is becoming the dominant contributor to total path delay [21, 22].

Table 2 shows layout statistics for the BSN and MBA chips. The densities of 15.0 kTX/mm^2 (thousands of transistors per mm^2) for the control-logic-dominated MBA chip and 77.9 kTX/mm^2 for the memory-dominated BSN chip are comparable to densities presented at the 1997 International Solid State Circuits Conference, e.g., 34.6 kTX/mm^2 in [23], 36.9 kTX/mm^2 in [24], or 25.4 kTX/mm^2 in [25].

The run times for placement and routing are very attractive considering that hardly any manual intervention is required. For example, a complete timing-driven layout for the MBA chip, consisting of two placement and optimization iterations and a routing step, can be performed in less than five days of processor time (**Table 3**).

Design for testability

For very complex VLSI chips, design for test (DFT) is required in order to achieve high test coverage and reasonable turnaround time. DFT consists of four major phases:

1. *Definition of test methodology and design of test macros*
   All of our designs follow the level-sensitive scan design (LSSD) rules [26]. This allows race-free testing and initialization of all memory elements in the chip at any level. The implementation is always full-scan. Our main test approach is built-in self-test (BIST), in which different state machines are designed that execute the test after initialization. BIST is used to test both combinational logic (LBIST) and memory arrays

(ABIST) [27, 28]. These tests can be used on all levels of hierarchy, from chip test in manufacturing through the power-on sequence in a customer's office. They can be run at system cycle speed in chip manufacturing using on-product clock generation and on-chip PLLs. A small number of special I/O circuits allow testing of all signal I/Os without contacting them at wafer test. This reduced-pin-count testing technique [29] allows the use of less expensive test vehicles in manufacturing for most of the tests. The custom library elements are checked early in the design phase for testability, and, if necessary, logic circuits are added to improve controllability and observability.

2. *Design of test control logic together with clock generation logic*
   Embedding test control logic into on-product clock generation allows more accurate testing by using the system clock distribution. The test control logic is very similar to the IEEE JTAG controller [30], but in addition it has several registers to set up and control the different tests that are executed. For example, registers are used to define the length of the LBIST test sequence, or the way of clocking, or to disable certain parts of the chip. The test control logic is designed once and reused on all chips in the set. The basic LSSD design and the common test controller are embedded in the functional portion of each chip in such a way that they are virtually invisible to the functional logic designer (**Figure 7**).
3. *Test structure verification (TSV)*
   An IBM-developed tool set, TestBench* [1], is used not only for test data generation, but also to check for design rule compliance. TestBench checks and analyzes compliance with LSSD and several other rules:
   - *Boundary scan rules* These rules enable us to test the I/O area independently of the internal logic of the chips, and vice versa. Our implementation [31] is similar to the IEEE JTAG boundary scan design.
   - *Self-test rules* In all self-test designs, propagation of undefined states into the signature analyzer is prohibited because it would corrupt the final signature. Another important check ensures that the self-test chain lengths are equal for all chips in the set, allowing us to reuse the same LBIST control logic on the chips.
   - *IDDQ rules* Because all of our chips are also tested with IDDQ test patterns, it is necessary that all current can be turned off for the measurements. We devoted a separate test I/O pin to control this.
4. *Testability analysis (TA)*
   The testability goal for our chips is 99.9% stuck-fault coverage and 95% delay-fault coverage. With TestBench we are able to generate the fault models as well as to analyze the problem areas. The implementation of LSSD full-scan in addition to LBIST enables TestBench to produce very high coverage almost immediately. The testability problem that we deal with is mainly redundancy removal. However, because our primary test method is LBIST, it is also very important to identify logic portions that are hard to test with random patterns. We add controllability and observability wherever possible to achieve at least 98% LBIST stuck-fault coverage.

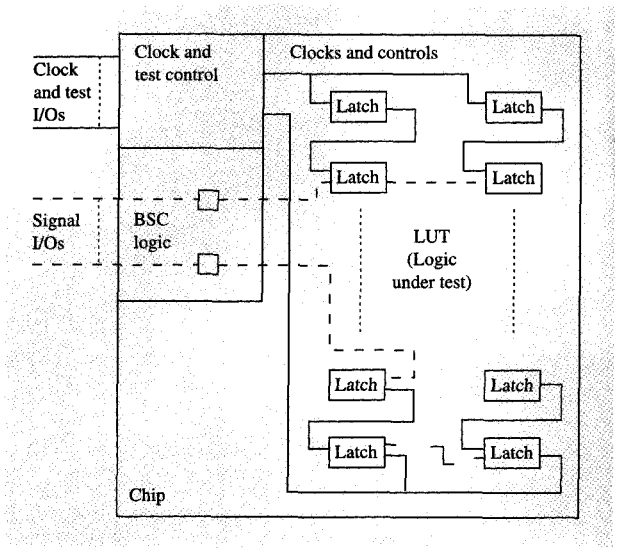

Figure 7
Test control logic.

Test pattern generation

Figure 8 shows the test pattern generation (TPG) flow. After a design has passed the TSV and TA checks, TPG generates the actual chip and/or module test data to be used during manufacturing, as well as the system LBIST signatures that are checked in the machine. TPG generates the following test data:

1. *Scan test*
   This test ensures the basic function of the implemented LSSD design and is key for any diagnostics performed in manufacturing.
2. *LBIST/ABIST test*
   The LBIST patterns include the initialization of the chip plus the calculated final signature, as well as intermediate signatures for debug and diagnosis. The ABIST patterns not only indicate success/failure but also identify failing array cells that can then be disabled and replaced by redundant array cells. The

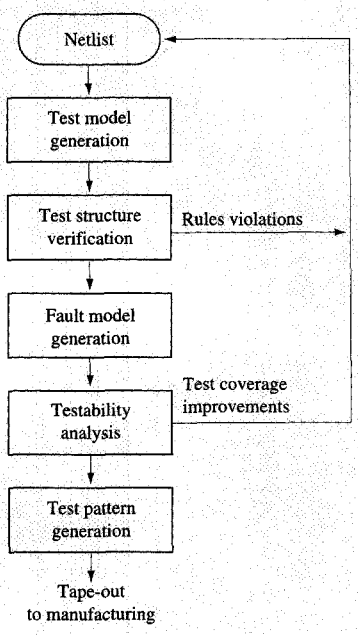

Figure 8

Test pattern generation flow.

LBIST/ABIST tests can be run in two different ways: controlled by a single oscillator using the on-product clock-generation logic, or controlled by dedicated tester clocks, the so-called LSSD clocks. This is important for diagnostic purposes.

3. *Deterministic test*
   Additional test patterns are generated to supplement the LBIST test coverage, in order to achieve 99.9% stuck-fault coverage.
4. *I/O test*
   Using the boundary-scan chain, all I/Os can be set independently to logic 1 or 0 so that these patterns are relatively easy to generate and very compact.

Table 4 shows TPG statistics for the MBA chip.

Outlook

We have described a standard-cell-based VLSI design system producing results which are competitive with custom design solutions in terms of density, in a very short turnaround time. In the future, we expect the logic complexity and the number of small custom macros to increase. To verify that this complexity can be handled by our methodology, we have successfully placed and routed an experimental design consisting of 580000 standard cells on a 1024-mm^2 image.

Table 4 TPG statistics for the MBA chip.

Test model gates	1.64×10^6
LSSD latches	95.5K
Stuck faults	5.3×10^6
Delay faults	4.8×10^6
LBIST	
Stuck-fault coverage (%)	98.93
CPU time on RS/6000 Model 590 (h)	25
Vectors	496K
Deterministic patterns	
Stuck-fault coverage (%)	99.83
CPU time on RS/6000 Model 590 (h)	1
Vectors	742

The low percentage of long nets inherent in our design approach should minimize the impact of the higher interconnection delay expected in future, denser chip technologies. We are currently focusing on improvements in parasitic extraction and the analysis and avoidance of crosstalk. Furthermore, efforts are being put into faster system-level simulation techniques.

*Trademark or registered trademark of International Business Machines Corporation.

References

1. *TestBench User's Guide, Version 2.1*, IBM Microelectronics Division, Endicott, NY 13760, 1995.
2. D. Brand, R. Damiano, L. van Ginneken, and A. Drumm, "In the Driver's Seat of BooleDozer," *Proceedings of the IEEE International Conference on Computer Aided Design (ICCAD)*, October 1994, pp. 518–521.
3. 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 J. Res. Develop.* **40,** No. 4, 407–430 (1996).
4. *EinsTimer User's Guide and Language Reference, Version 1.3*, IBM Microelectronics Division, Hopewell Junction, NY 12533, 1995.
5. G. Doettling, K. J. Getzlaff, B. Leppla, W. Lipponer, T. Pflueger, T. Schlipf, D. Schmunkamp, and U. Wille, "S/390 Parallel Enterprise Server Generation 3: A Balanced System and Cache Structure," *IBM J. Res. Develop.* **41,** No. 4/5, 405–428 (1997, this issue).
6. IBM: "CMOS 5X 2.5V Gate Array/Standard Cell," *http://www.chips.ibm.com/products/asics/tech/cmos5x/cmos5x.html.*
7. C. W. Koburger III, W. F. Clark, J. W. Adkisson, E. Adler, P. E. Bakeman, A. S. Bergendahl, A. B. Botula, W. Chang, B. Davari, J. H. Givens, H. H. Hansen, S. J. Holmes, D. V. Horak, C. H. Lam, J. B. Lasky, S. E. Luce, R. W. Mann, G. L. Miles, J. S. Nakos, E. J. Nowak, G. Shahidi, Y. Taur, F. R. White, and M. R. Wordeman, "A Half-Micron CMOS Logic Generation," *IBM J. Res. Develop.* **39,** No. 1/2, 215–227 (1995).
8. M. Suzuki, N. Ohkubo, T. Yamanaka, A. Shimizu, and K. Sasaki, "A 1.5ns 32b CMOS ALU in Double Pass-Transistor Logic," *Proceedings of the International Solid State Circuits Conference*, 1993, pp. 90–91.
9. W. Roesner, "A Hardware Design Language for Logic Simulation and Synthesis in VLSI," *Proceedings of IEEE COMPEURO*, May 1987, pp. 311–314.

10. *Design Entry: Composer User Guide*, Cadence Design Systems Inc., 555 River Oaks Parkway, San Jose, CA 95134, 1994.
11. A. Kuehlmann, A. Srinivasan, and D. P. LaPotin, "Verity—A Formal Verification Program for Custom CMOS Circuits," *IBM J. Res. Develop.* **39,** No. 1/2, 149–165 (1995).
12. W. Roesner, "A Mixed Level Simulation System for VLSI Logic Designs," *Proceedings of IEEE COMPEURO*, May 1987, pp. 196–199.
13. W. G. Spruth, *The Design of a Microprocessor*, Springer-Verlag, Berlin, 1989.
14. J. Vygen, "Algorithms for Large-Scale Flat Placement," to be published in *Proceedings of the 34th Design Automation Conference*, 1997.
15. J. Vygen, "Algorithms for Detailed Placement of Standard Cells," to be published in *Proceedings of Euro-DAC 1997.*
16. R.-S. Tsay, "An Exact Zero-Skew Clock Routing Algorithm," *IEEE Trans. Computer-Aided Design* **14,** No. 12, 242–249 (February 1993).
17. K. D. Boese and A. B. Kang, "Zero-Skew Clock Routing Trees with Minimum Wirelength," *Proceedings of the ASIC Conference*, 1992, pp. 17–21.
18. U. Fassnacht and J. Schietke, "Timing Analysis and Optimization of a High Performance CMOS Processor Chipset," to be published in *Proceedings of Euro-DAC 1997.*
19. A. Hetzel, "A Sequential Detailed Router for Huge Grid Graphs," to be published in *Proceedings of Euro-DAC 1997.*
20. T. Stoehr, M. Alt, A. Hetzel, and J. Koehl, "Analysis, Reduction and Avoidance of Crosstalk on VLSI Chips," to be published in *Proceedings of Euro-DAC 1997.*
21. D. J. Hathaway, R. R. Habra, E. C. Schanzenbach, and S. J. Rothman, "Circuit Placement, Chip Optimization, and Wire Routing for IBM IC Technology," *IBM J. Res. Develop.* **40,** No. 4, 453–460 (1996).
22. K. Sato, M. Kawarabayashi, H. Emura, and N. Maeda, "Post-Layout Optimization for Deep Submicron Design," *Proceedings of the 33rd Design Automation Conference*, 1996, pp. 740–745.
23. D. Greenhill, E. Anderson, J. Bauman, A. Charnas, R. Cheerla, H. Chen, M. Doreswamy, P. Ferolito, S. Gopaladhine, K. Ho, W. Hsu, P. Kongetira, R. Melanson, V. Reddy, R. Salem, H. Sathianathan, S. Shah, K. Shin, C. Srivatsa, and R. Weisenbach, "A 330 MHz 4-Way Superscalar Microprocessor," *Proceedings of the International Solid State Circuits Conference*, 1997, pp. 166–167.
24. M. R. Choudhury and J. S. Miller, "A 300 MHz CMOS Microprocessor with Multi-Media Technology," *Proceedings of the International Solid State Circuits Conference*, 1997, pp. 170–171.
25. A. K. Jain, R. P. Preston, P. J. Bannon, M. S. Bertone, R. P. Blake-Campos, G. A. Bouchard, D. S. Brasili, D. A. Carlson, R. W. Castelino, K. M. Clark, S. Kobayashi, B. P. Lilly, S. Mehta, B. S. Miller, R. O. Mueller, A. Olesin, Y. Saito, and V. Yalala, "1.38cm^2 550MHz Microprocessor with Multimedia Extensions," *Proceedings of the International Solid State Circuits Conference*, 1997, pp. 174–175.
26. E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testability," *Proceedings of the 14th Design Automation Conference*, 1977, pp. 462–468.
27. Cordt W. Starke, "Design for Testability and Diagnosis in a VLSI CMOS System/370 Processor," *IBM J. Res. Develop.* **34,** No. 2/3, 355–362 (1990).
28. R. Ihle and C. W. Starke, "Array Built-In Selftest for a S/390 Microprocessor Chipset," *Proceedings of the European Test Conference*, 1997, pp. 97–100.
29. R. W. Bassett, M. E. Turner, J. H. Panner, P. S. Gillis, S. F. Oakland, and D. W. Stout, "Boundary-Scan Design Principles for Efficient LSSD ASIC Testing," *IBM J. Res. Develop.* **34**, No. 2/3, 339–354 (1990).
30. "IEEE Standard Test Access Port and Boundary-Scan Architecture," IEEE Standard 1149.1-1990, IEEE Standards Board, 345 E. 47th St., New York, NY 10017, 1990.
31. P. S. Gillis, U. Baur, K. McCauley, and F. Woytowich, "Delay Test of Chip I/Os Using LSSD Boundary Scan," to be published in *Proceedings of the International Test Conference*, 1997.

Received January 16, 1997; accepted for publication May 5, 1997

Bernhard Kick *IBM Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (KICK at BOEVM3).*
Mr. Kick is an Advisory Engineer in the S/390 Processor Development group. In 1984 he received an M.S. degree in electrical engineering from the Technical University of Munich, Germany. He joined IBM in 1986 and has been working on logic synthesis and verification.

Ulrich Baur *IBM Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (BUBA at BOEVM4).*
Mr. Baur is an Advisory Engineer in the S/390 Processor Development group. He received an M.S. degree in electrical engineering from the University of Stuttgart, Germany, in 1984, joining IBM the same year. He has been working on design for testing and testing data generation.

Juergen Koehl *IBM Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (KOEHL at BOEVM4).*
Dr. Koehl is an Advisory Engineer in the VLSI Design Centre. He received his Ph.D. in mathematics from Bonn University in 1987; his Ph.D. thesis was on algebraic cycles on Hilbert modular surfaces. From 1987 to 1989 he was a Research Staff Member at the Institute for Discrete Mathematics in combinatorial optimization for VLSI design. He joined IBM in 1989 and is responsible for physical design methodology. Dr. Koehl is Chairman of the ITG/GI/GMM Fachgruppe "Layout" and a senior member of the IEEE.

Thomas Ludwig *IBM Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (TLUDWIG at BOEVM4).* Mr. Ludwig received his M.S. degree in electrical engineering from the Technical University of Berlin in 1984 and joined IBM the same year. He has been working on circuit design of custom logic macros and their design methodology at the VLSI Logic Chip Development Department in Boeblingen. Mr. Ludwig is a member of the IEEE Circuits and Systems Society.

Thomas Pflueger *IBM Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (PFLUEGER at BOEVM3).* Mr. Pflueger is an Advisory Engineer in the S/390 Processor Development group. He received an M.S. degree in electrical engineering from the Technical University of Munich, Germany, in 1983, joining IBM the same year. Mr. Pflueger has been working on processor logic design.