A Flat, Timing-Driven Design System for a High-Performance CMOS Processor Chipset

Juergen Koehl, Ulrich Baur, Thomas Ludwig, Bernhard Kick, Thomas Pflueger
IBM Entwicklung GmbH Boeblingen, Schoenaicher Str. 220
71032 Boeblingen, Germany
koehl@de.ibm.com

Abstract

We describe the methodology used for the design of the CMOS processor chipset used in the IBM S/390 Parallel Enterprise Server - Generation 3. The majority of the logic is implemented by standard cell elements placed and routed flat, using timing-driven techniques. The result is a globally optimized solution without artificial floorplan boundaries. We will show that the density in terms of transistors per mm^2 is comparable to the most advanced custom designs and that the impact of interconnect delay on the cycle time is very small. Compared to custom design, this approach offers excellent turn-around-time and considerably reduces overall effort.

1: Introduction and Motivation

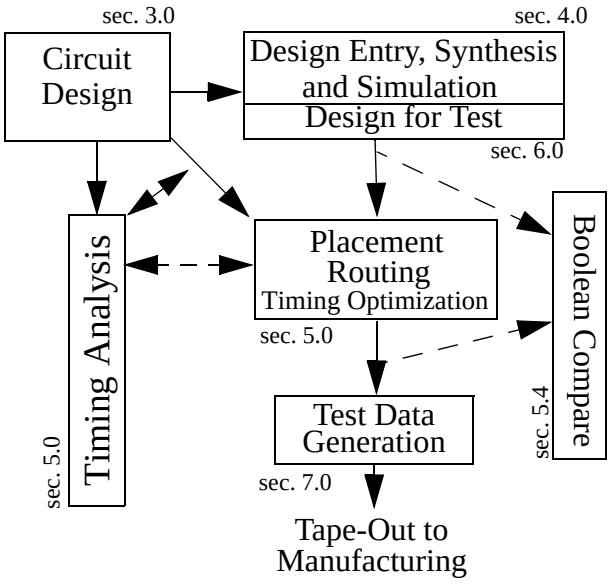
Custom design is the dominant design style for high performance processors. The physical partitions used in floorplanning are typically identical to the logical partitions. The sub-optimality caused by the floorplan boundaries is reduced by changing the functional partitions based on back annotation of floorplanning results. This however, adds additional turn-around-time. The advantage offered by a custom approach is that different design styles, ranging from full-custom layout to standard cell design, may be used on individual partitions. In this sense, custom design focuses on local optimization at the expense of global optimization.

A standard cell design approach makes it possible to globally apply advanced optimization algorithms on the entire design, thus improving the quality of the layout while significantly reducing manual effort. The use of basic standard cell elements reduces the complexity to the extent that the entire design can be handled flat by layout and test data generation tools, removing the need for artificial floorplan boundaries.

While our approach does use custom memory arrays and some small custom logic macros for special functions, it is important to note that these logic macros are small enough to be placed under global timing optimization criteria and do not need any floorplanning or preplacement. The majority of the combinatorial logic, however, is implemented in standard cells, and is placed and wired flat with the primary objective being cycle time reduction. A full timing-driven layout of the PU chip with its 164,000 placeable objects and 600,000 connections can be performed in less than 5 days!

Logic entry, synthesis, and simulation are performed based on functional units. The fact that the logic partitioning is not used during layout removes the need to optimize this partitioning based on timing and layout considerations. Time consuming logic repartitioning based on layout back annotation is no longer needed.

FIGURE 1. Design Flow



The testing methodology includes Design for Test

(DFT) to assure a high test coverage, and test-pattern generation (TPG) to enable testing, analysis, and debugging of chips in manufacturing. Key are fast turn-around-time and high-quality testing.

Test data generation, circuit and logic design, as well as layout verification, are performed with IBM internal tools. The layout optimization tools were developed at the Institute for Discrete Mathematics at Bonn, Germany, in close cooperation with IBM Boeblingen. This cooperation minimizes the lead time required to incorporate combinatorial optimization research results into production tools. The availability of these advanced methods in combinatorial optimization enables us to apply the methodology outlined in this paper to designs with more than 500,000 placeable objects and 1,700,000 connections.

This paper is intended to give an overview of the processor chipset and the methodology used (see Figure 1), and to set a frame for the presentation of the mathematical algorithms used in layout and timing optimization ([3], [5] and [12]) which were key to the implementation of this methodology.

The methodology has been used successfully on a S/390 CMOS processor chipset which is the heart of the S/390 Parallel Enterprise Server - Generation 3 which has been on the market since September of 1996.

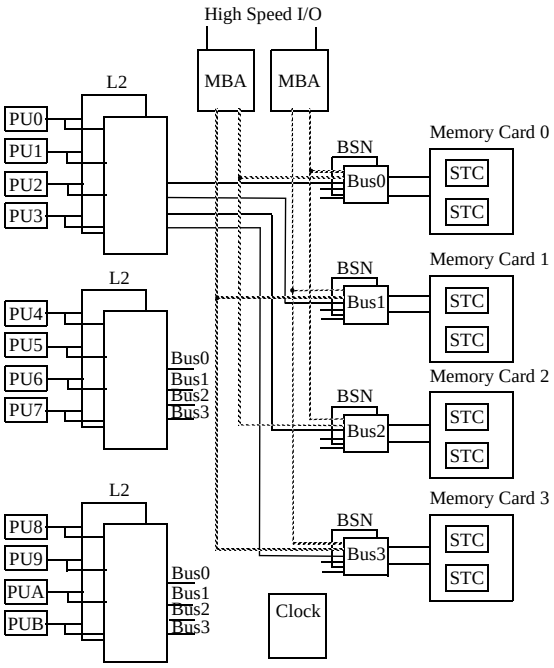
2: System Overview

2.1: Chipset

The chipset (Figure 2) consists of a clock chip, processing units (PU), level 2 caches (L2), bus switching network adapters (BSN) with level 3 cache, two I/O controllers (MBA) and storage controllers (STC). A tightly coupled S/390-multiprocessing system with up to 12 processors can be built with this chipset, and can address a physical storage of 16 Gbytes. Each PU is connected to the memory through L2 and BSN by four 16-byte-wide buses supporting 2-way interleaving each. This results in a peak bandwidth of more than 10 GB per second for the entire chipset. The system runs with a cycle time of 5.9 ns.

The chipset performs substantial error checking to increase resistance to soft errors. All data path elements are parity checked, and for large SRAM devices soft error recovery or ECC is used. Soft error recovery takes advantage of the fact that data is often held a second time in the system, e.g. in the cache and in memory, and in the event of a parity error, the duplicated data are automatically copied to the faulty device. ECC provides 1-bit failure correction and 2-bit failure detection.

FIGURE 2. System Overview (max. configuration)



2.2: Cache Structure

The chipset implements a three-level cache hierarchy. The level 1 (L1) and level 2 (L2) caches are private for each PU. The L3 cache is a system-wide shared cache. The line size for all caches is 128 bytes. Eight cycles, plus the latency cycles described in Table 1, are needed to transfer a complete cache line.

The L1 caches, located on the PU chips, are 32-kbyte 8-way set-associative write-through, and can be accessed within one cycle. The L2 caches, which are 256-kbyte 8-way set-associative write-in, are implemented as separate chips. The L2s implement the MESI protocol to ensure system-wide data consistency between all L1s and L2s. The 2 Mbytes L3 cache is located on the BSN chips. This shared cache is write-through, and designed for high-speed PU-L3 communication.

TABLE 1. Memory Hierarchy

	Size	Associativity	Latency cycles
L1	32 KB	8	0
L2	256 KB	8	4
L3	2 MB	8	13
Memory	8 GB	-	32

2.3: Processing Unit Chip

The PU implements a 32-bit CISC data architecture which executes the S/390 instruction set. The main parts of this chip are the L1-cache, floating point, address generation, ALU, shift unit, register file, timers, and data compression unit. Data compression is fully implemented in hardware. The main floating point data flow consists of an add-subtract flow with 116-bit width, and a signed multiplier with 60*60 bits. The pipeline structure for all non-floating-point op-codes runs up to four stages deep:

- instruction fetch
- decode
- execute
- putaway

In case of floating-point computational operation, the four stages above are expanded to eight stages. Therefore the most frequently used PU operations, and all floating-point operations except divide, square root, and extended multiply, require only one execution cycle. All floating-point and 108 PU-related instructions are executed by hardware only, as in RISC processors. The remaining PU instructions are executed via a vertical micro code. A portion of this micro code is loaded on system start to a 32-kbyte PU on-chip SRAM device. Another part of the micro code is held in a 32-kbyte on-chip ROM, and the remaining portion has to be transferred from the memory through STC, BSN, and L2 if needed. A micro code transfer unit consist of 128 bytes. In a commercial environment, this instruction implementation leads to an infinite cache cycles-per-instruction (cpi) performance of 2.4.

3: Technology and Circuit Design

3.1: Technology

The CMOS process used [10] on this chipset was developed by IBM Microelectronics Division. The technology provides six layers of metallization, one for internal circuit wiring only, and four layers for wiring in a 1.8µm wiring pitch. The last metal layer is used primarily for wiring redistribution to the C4 array of signals and supplies

TABLE 2. Technology

Feature	Value
Supply Voltage	2.5 V
Leff	0.25 µm
Minimum Feature Size	0.33 µm
Tox	7 nm
Metal layers	1 + 5

3.2: Library and Chip Image

The standard cell library provides a set of logic gates, latches, and I/O's. They fit into 3.5 million legal placement locations and are interconnected through the x/y-wiring tracks defined in the chip image. The I/O-cells are allowed to be placed anywhere among the 3.5 million legal locations.

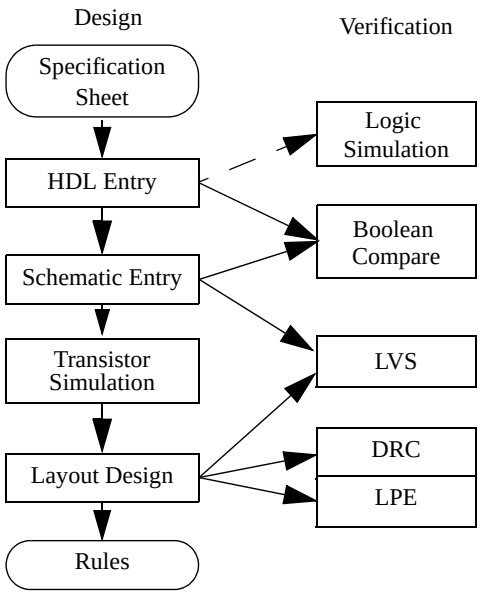
3.3: Custom Circuit Design

The base standard cell library provides simple logic gates, but for the special needs of a S/390 processor a small set of custom Logic Macros and custom SRAM Macros was required. The custom implementation of the macros gives the circuit designer the freedom to use special circuit design techniques, like dynamic and double pass [11] circuits to improve the propagation delay.

The circuit design flow begins with a specification sheet defining the macro requirements. With this information an HDL model in a proprietary hardware description language [7] is built. This model of the macro defines the logic behavior and is 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 implementation of the logic function in transistors with a schematic entry tool. 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.

FIGURE 3. Circuit Design Flow



A boolean equivalence checker [6] compares the transistor schematics against the “golden model” and gives early simulation-independent feedback of the correct implementation.

An early timing rule is generated for the chip-level delay-calculator [3]. This early rule will be replaced later in the design process by the final rule, based on layout-extracted circuit information.

The device and net information in the schematics is used by the 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 usage of circuit generators for base logic functions like NANDs. The custom macros can be used like big standard cell books, placeable in any legal location.

The custom macro layout is fed into the 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 re-simulation to make sure that the function and performance are still correct. This netlist is the source for the final timing rule representing the 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 rule 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 rule is verified against the “golden” HDL model to guarantee logic equivalence between both implementations.

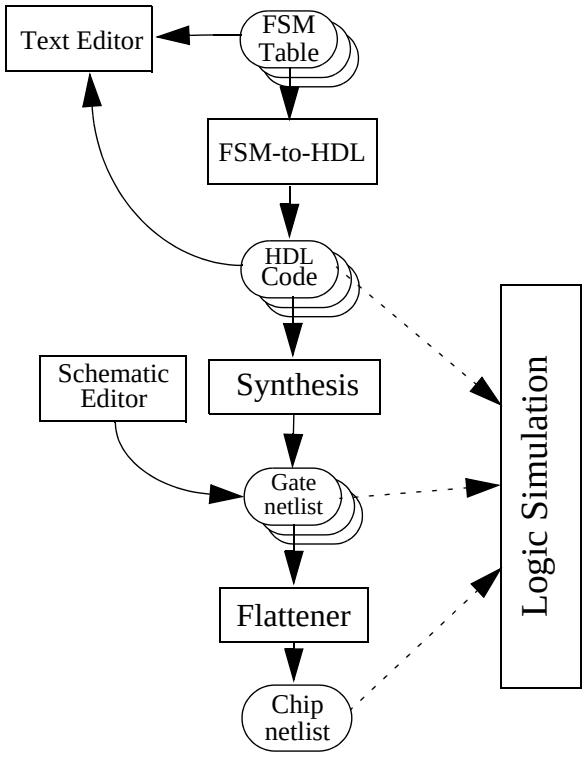
4: Design Entry, Synthesis, and Simulation

4.1: Design Entry

The design system accepts design data in three forms: gate level schematics, hardware design language (HDL) code (see [7]), and finite state machine (FSM) tables; see Figure 4.

Gate level schematics are preferred for data flow dominated designs, whereas HDL code and FSM tables are well suited for control flow dominated designs. Most parts of the I/O chips are HDL code or FSM table designs. The level of description is similar to the concurrent subset of VHDL: boolean expressions, signal assignments, component instantiations, etc..

FIGURE 4. Logic Design Data Flow



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.

4.2: Logic Synthesis

The logic synthesis system, BooleDozer [1], 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. Synthesis uses the same delay calculator as placement and routing, with the exception that interconnect capacitances and resistances are estimated as a function of fanout, based on statistics from placement and routing.

As a full chip design cannot be synthesized in one run, it has to 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 turn-around times. Synthesis times range from one to ten CPU hours per partition, resulting in over-night turn-around if only a few partitions are synthesized. Turn-around time for

a complete chip is two to three days.

Partitioning requires that delay objectives for the chip be broken down into delay objectives for each partition. This process, called slack apportionment, assigns delay objectives to partitions in such a way that if each partition meets its delay objective, then the chip also meets the delay objective.

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.

4.3: Simulation

Extensive logic simulation at the functional unit, chip, and system level is performed to verify the functional correctness of the designs ([8], [9]). Simulation is performed without regard to delay, leaving timing verification to the delay calculator [3]. This approach nicely separates timing aspects from functional aspects and speeds up simulation considerably.

Unit level and chip level simulation are performed by the logic design groups, primarily using HDL models and test cases developed by the logic designers. This mode of simulation is used mainly in the early stages of the design to fix the easy bugs.

The bulk of simulation happens 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.

The tests performed include micro-code tests and architecture tests. Micro-code tests are low level tests that simulate at the hardware implementation (micro-code execution) level. They are loaded into, and executed from, the control store of the processor. Architecture tests are assembly language programs that check that the hardware works according to the S/390 specification. They are loaded and executed from main memory, and require correct cache and memory models, and S/390 micro-code.

Overall the test set contains about 10,000 self-checking test cases, of which a few thousand were developed and hand-coded over the last decade. The remaining tests were generated, or are generated on the fly during simulation, by various weighted random pattern test generators. A couple of weeks is needed to completely simulate this test set of more than 100 million machine cycles.

Upon completion of unit simulation, system simulation begins, and may be performed in parallel with logic design. A cluster of 46 S/390 systems, consisting of 268 processors is dedicated to simulation.

Building a new simulation model after a design change, and restarting the cluster of S/390 systems with the new model, takes about half a day. Overall more than 300 million cycles were simulated on various versions of the model over a period of 6 months.

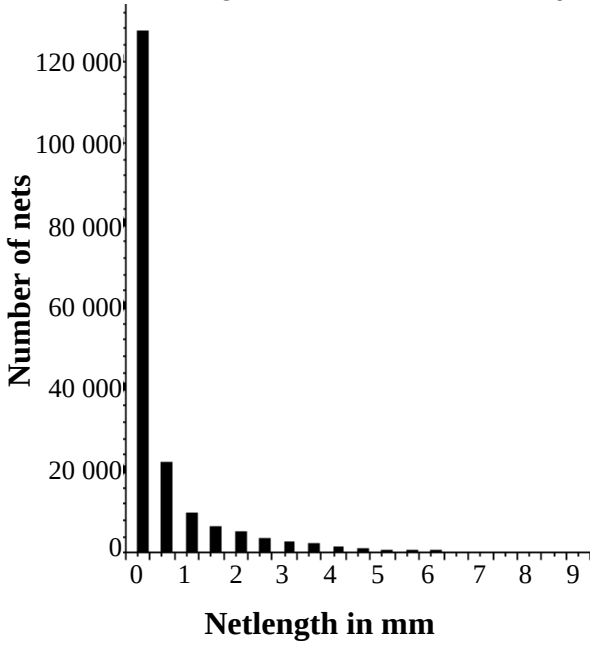
5: Flat, Timing-Driven Layout

Our chip place and route tools have been so refined over the course of four processor generations, that pure routeability is no longer our major concern. This allows us to focus fully on cycle time reduction during the place/route/optimization phase. To be able to judge the quality of a given layout, we need a reasonable lower bound for the possible cycle time.

A natural lower bound can be obtained by a static timing analysis of the logic network assuming a netlength of zero for each net. In other words: each circuit drives the input capacitances of the next stage with interconnect length set to zero. Using different design approaches such as floorplanning and timing-driven placement, we have compared the actual post-routing cycle time to this hypothetical “zero-netlength” cycle time. The design approach that consistently produces the lowest delay directly attributable to interconnect, is non-floorplanned (flat) and timing-driven. With this approach the final cycle time is consistently within 15% of the cycle time based on a zero-netlength timing analysis. An interesting observation is that the highest increase is obtained by the **flat non**-timing-driven approach and that the floorplanned approaches are in between. This shows that the flat approach is only superior if a global timing-driven layout methodology is used.

Our experience does not support the common industry view that “interconnect delay represents the majority of delays within a design”.

TABLE 3. Netlength Distribution for PU Chip



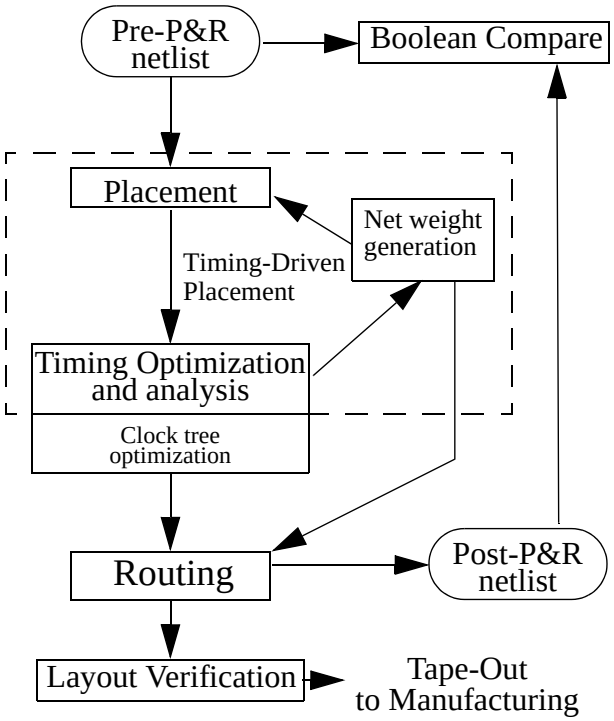
On a 14.6 mm ❑ chip, about 70% of the nets are below 0.5 mm and there are very few nets above 5 mm in length (see Table 3). Restricting the functional units to floorplan regions introduces global nets, which are typically longer. This relatively small interconnect delay adder is an inherent advantage of our design system. It will show its merits even more in future, denser technologies with higher RC delay adder.

The layout flow is shown in Figure 5, we will briefly describe the main steps.

5.1: Placement

The possibility to place and route complex designs flat, timing-driven is hence an important prerequisite for the design methodology presented here. It is made possible by quadratic optimization combined with a new quadrisection approach, developed by J. Vygen (see [12]). The timing-driven approach is based on netweights, that are generated based on a timing analysis and optimization of a non-timing-driven placement. High weights are only assigned to nets with timing violations, that cannot be solved by power level optimization or buffer insertion as described below. A branch-and-bound approach used in macro placement supports the placement of custom macros and standard cells in a single run without any preplacement. A description of the detailed placement can be found in [13].

FIGURE 5. Layout Flow



5.2: Timing Analysis and Optimization

As synthesis is done without knowledge of the actual interconnect length, in-place optimization techniques are used to further improve the cycle time. This is done in three steps:

1. Clock synthesis

   is performed following placement. The clock tree is not considered during placement but instead re-synthesized after placement based on a zero-skew approach. Routing information for balanced routing is created as an input to the routing step.

2. Power level optimization

   is performed for timing optimization and power reduction. It uses one of the 5 driving capabilities available for each standard cell circuit.

3. Buffer insertion

   is performed in conjunction with power level optimization on failing paths that could not be solved by power level optimization alone. The decisions of this algorithm are always based on actual placement data as each circuit added is assigned to a placement location.

   Details on timing analysis and optimization techniques can be found in U. Fassnacht and J. Schietke in [3].

5.3: 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 subsequent 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 sub-optimality introduced by the necessary pin propagation in a hierarchical approach.

The routing tool supports different wire widths and spacing restrictions. A crosstalk analysis and crosstalk violation removal capability has been implemented. The memory efficiency of the routing approach can be seen in Table 5, details will be presented by A. Hetzel in [5].

5.4: Boolean Compare and Engineering Changes

To avoid any risk of introducing logic errors during in-place optimization, a Boolean Equivalence checking tool [6] is used to verify the equivalence of the pre- and post-layout netlist.

Late metallization-only changes are supported either by re-routing or by the use of gate array books. This process is complicated by the fact that in-place optimization during lay-

out, and late functional changes by the logic designers, are done concurrently. We have implemented a flow to incorporate the functional changes performed on the pre-layout netlist into the post-layout netlist.

5.5: Results

TABLE 4. Design Statistics

	PU	L2	MBA
Chip-Size	213 mm^2	269 mm^2	240 mm^2
Standard cells	164 k	87 k	206 k
Pins	616 k	339 k	771 k
Nets	184 k	104 k	
Custom macros	98	341	89
Transistors	7.2 M	17.9 M	3.6 M
Global signal netlength	126 m	90 m	122 m
Signal I/O's	744	928	770
Cycle time	5.9 ns	5.9 ns	5.9 ns

The density of 33.8 k TX's per mm^2 is comparable to the density achieved by recently announced state of the art custom design microprocessors present at ISSCC 1997 (e.g. 34.6 TX's/mm^2 in [4] or 36.9 k TX's/mm^2 in [2]).

FIGURE 6. MONPU Chip Colored by Partition

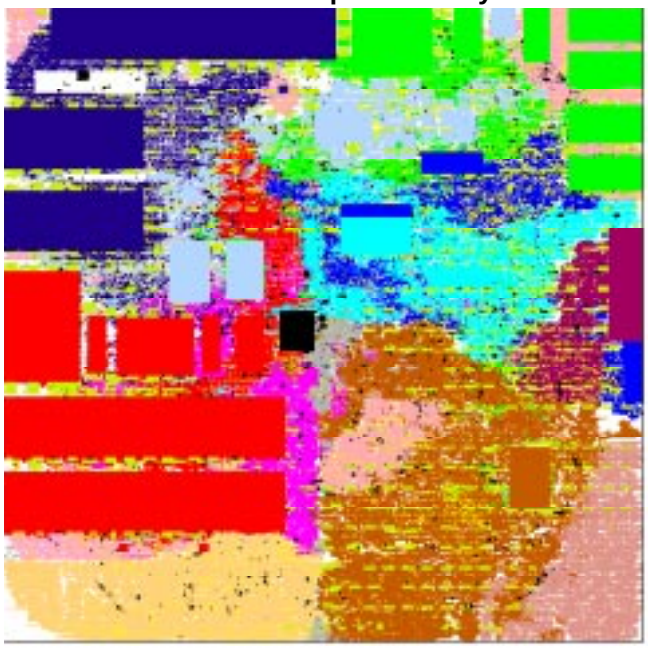


TABLE 5. Memory and Run Times

Step	Run times on RS/6000 590	Memory
Placement	12 h	500 MB
In-place optimization	21 h	1000 MB
Routing	21 h	220 MB

6: Design for Test

For very complex VLSI chips, Design for Test (DFT) is absolutely required in order to achieve high test coverage and excellent TAT.

DFT consists of four major phases:

1. Definition of test methodology and design of test macros:

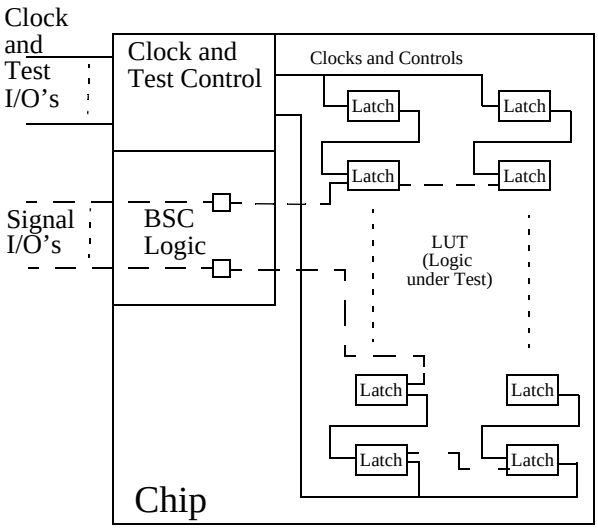
   All our designs follow the LSSD (level sensitive scan design) rules. This allows race-free testing and initialization of all memory elements in the chip at any level. The implementation is always full scan. Our main test approach is built-in-selftest (BIST), where different state machines are designed that execute the test after the initialization. BIST is used to test both logic (LBIST) and arrays (ABIST). Early in the design phase, macros are designed such that a common methodology is used across all chips in the machine. Our LBIST and ABIST tests can be used on all levels of hierarchy, from chip test in manufacturing all way up to the power-on sequence in the customers office. The LBIST implementation follows the STUMPS approach. These tests can be performed at system cycle speed in chip manufacturing using SGC (Self Generated Clocks) and on-chip PLLs. Special I/O books were designed that allow testing of the signal I/Os without contacting them at wafer test. This is also called RPCT (reduced pin count testing) and allows the use of less expensive test equipment in manufacturing for most of the tests. The more complex library elements are checked early in the design phase for testability, and if necessary, logic is added to improve controllability and observability.

2. Checking for design rule compliance (TSV, test structure verification):

   Compliance with the LSSD rules is checked using an IBM-developed tool set called TestBench. TestBench not only supports TSV, but is also used to generate all test patterns. In addition to the LSSD rules, several other rules are checked and analyzed:

a. *BSC (boundary scan) rules*: These rules enable us to test the I/O area independent of the internal logic of the chips, and vice versa. The implementation is similar to the JTAG boundary scan design.

FIGURE 7. Test Control Logic



b. *Selftest rules*: In all selftest designs, propagation of X-states into the signature analyzer is prohibited, because it corrupts the final signature. Another important check is the common-defined selftest chain-length, allowing us to use the same LBIST control logic across all chips.

c. *IDDQ rules*: All our chips are also tested with IDDQ test patterns. To enable this, it is necessary that all current can be turned off for the measurements. We spent a separate test I/O to control this.

3. Design of test control logic together with clock generation logic:

   Embedding test control logic into the clock generation logic allows more accurate testing by using the system clock distribution. The control logic is very similar to the JTAG controller defined by IEEE, but in addition, it has several test registers to setup and control the different tests that are executed. For example, registers are used to define the length of the LBIST test sequence, the way of clocking, or just to enable certain parts of the chip. The test logic is designed only once for all chips and is embedded such that it is virtually invisible to the logic designer. Both the basic LSSD design and the common design for the test controller are easily merged with the logic. Figure 7 is a high-level view of the logic interfaces.

4. Testability analysis (TA):

   The testability goal for our chip designs is 99.9% DC fault coverage (using the stuck-at fault model) and 95% AC fault coverage (using the transition fault model). With TestBench we are able to generate the fault models as well as to analyze the problem areas. The implementation of LSSD full scan enables TestBench to produce very high coverage almost immediately. The testability problem that we deal with is mainly redundancy removal. Because our main test method is LBIST, testability analysis for random resistant logic is also very important. To improve this test, we must add controllability and observability wherever possible. Our goal here is to achieve 98% DC fault coverage with LBIST only.

7: Testpattern Generation

Figure 8 describes the test pattern generation flow. After the steps TSV and TA (described above) follows the generation of the actual test data which is used during manufacturing for chip and/or module testing, as well as the generation of the system LBIST signatures that are checked in the machine. TPG generates the following tests:

1. Scantest:

   This test insures the basic function of the implemented LSSD design and is key for any diagnostics done in manufacturing.

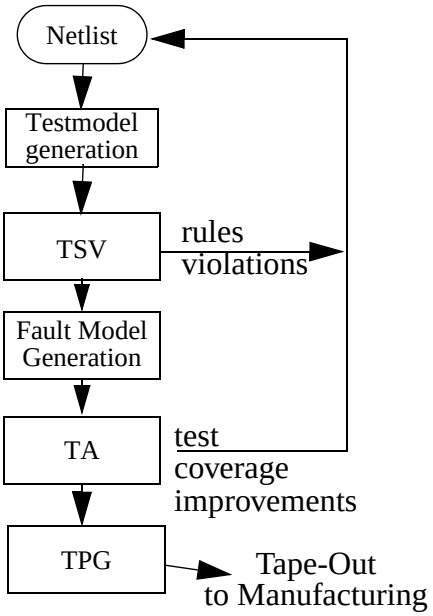
2. LBIST/ABIST test:

   For LBIST, the generated test patterns include the initialization of the chip plus the calculated signature, as well as intermediate signatures for debug and diagnosis. The ABIST patterns are very similar to the LBIST patterns. After the state machine has stopped, registers not only indicate success/failure but also contain the fuse information if redundancy is built into the arrays. We have the capability to run these tests in two different ways: using the SGC logic, the tests are run by an oscillator only, but we also have the capability to run them with dedicated tester clocks, the so called LSSD clocks. This again is very important for diagnostic purposes.

3. Deterministic test:

   Additional testpatterns are generated to supplement the LBIST test coverage, in order to achieve 99.9% DC fault coverage.

FIGURE 8. Testpattern Generation (TPG) Flow



4. I/O test:

   Using the BSC implementation, the I/O tests are very easy to generate. With the boundary scan chain we can setup all I/O's independently to '1' or '0' so that these patterns are very compact. Only 1% to 3% of the chip logic needs to be simulated.

   Refer to Table 6 for TPD statistics.

TABLE 6. Test Data Statistics

Testmodel books	950 k
LSSD latches	46 k
Faults DC	2.8 M
Faults AC	3.2 M
LBIST:	
- DC coverage	96.70%
- CPU time on RS6000/590	12 h
- Vectors	500 k
Deterministic Patterns:	
- DC coverage	99.89%
- CPU time on RS6000/590	5 h
- Vectors	1700

8: Outlook

We have presented a flat, timing-driven VLSI design system, producing results which are very competitive with custom design solutions in both density and clock frequency.

To verify that future complexities can be handled by our methodology, we have successfully placed and routed an experimental design consisting of 580.000 placeable objects on a 1024 mm^2 image.

The low percentage of long nets inherent in our design style will minimize the impact of the higher interconnect delay expected in future, denser, technologies. Together with our partners from Bonn University, we are currently testing and implementing techniques for global transistor sizing for further cycle time reduction and power minimization.

Considerable effort is being put into faster system level simulation techniques and their implications to our logic design style.

9: References

[1] D. Brand, R. Damiano, L. van Ginneken, A. Drumm: In the Driver's Seat of BooleDozer. *Proc. ICCAD, October 1994*, pp. 518-521.

[2] Choudhury, Miller: A 300 MHz CMOS Microprocessor with Multi-Media Technology. Proc. of ISSCC 1997, pp. 170-171.

[3] U.Fassnacht, J.Schietke: Timing Analysis and Optimization of a High-Performance CMOS Processor Chipset. *To appear in Proc. of DATE 1998.*

[4] Greenhill et. al.: A 330MHz 4-Way Superscalar Microprocessor. Proc. of ISSCC 1997, pp. 166-167

[5] A. Hetzel: A Sequential Detailed Router for Huge Grid Graphs. *To appear in Proc. of DATE 1998.*

[6] A. Kuehlmann et. al.: Verity - A Formal Verification Program for Custom CMOS Circuits. *IBM Journal of Research and Development. Vol. 39, No.1/2.*

[7] W. Roesner: A Hardware Design Language for Logic Simulation And Synthesis in VLSI. *Proc. IEEE COMPEURO, May 1987*, pp. 311-314.

[8] W. Roesner: A Mixed Level Simulation System for VLSI Logic Designs. *Proc. IEEE COMPEURO, May 1987*, pp. 196-199.

[9] W. G. Spruth: The Design of a Microprocessor. *Springer-Verlag, 1989.*

[10] IBM: CMOS5X 2.5V Gate Array/Standard Cell. "http://www.chips.ibm.com/products/asics/tech/cmos5x/cmos5x.html"

[11] Suzuki et. al.: A 1.5ns 32b CMOS ALU in Double Pass-Transistor Logic. *Proc. ISSCC '93, pp. 90-91.*

[12] J.Vygen: Algorithms for Large-Scale Flat Placement. *Proc. of the 34th Design Automation Conference, 1997, pp.746-751.*

[13] J.Vygen: Algorithms for Detailed Placement of Standard Cells. *To appear in Proc. of DATE 1998.*