The circuit and physical design of the POWER4 microprocessor

by J. D. Warnock
J. M. Keaty
J. Petrovick
J. G. Clabes
C. J. Kircher
B. L. Krauter
P. J. Restle
B. A. Zoric
C. J. Anderson

The IBM POWER4 processor is a 174-million-transistor chip that runs at a clock frequency of greater than 1.3 GHz. It contains two microprocessor cores, high-speed buses, and an on-chip memory subsystem. The complexity and size of POWER4, together with its high operating frequency, presented a number of significant challenges for its multi-site design team. This paper describes the circuit and physical design of POWER4 and gives results that were achieved. Emphasis is placed on aspects of the design methodology, clock distribution, circuits, power, integration, and timing that enabled the design team to meet the project goals and to complete the design on schedule.

Introduction

The POWER4 chip provides the processing power for eServer p690, the recently introduced high-end, IBM 64-bit POWER-architecture, 8-to-32-way server system [1]. The chip, shown in **Figure 1**, includes two microprocessors, 1.44 MB of shared L2 cache memory plus the directory for a 32MB off-chip cache, a 500^{+}-MHz interconnection fabric, high-bandwidth buses and I/O designed to allow building an eight-way system on a single multi-chip module, and the logic needed to support large SMPs [2]. The microprocessor core is an out-of-order, speculative, eight-issue superscalar design containing eight execution units, a 64KB L1 instruction cache, and a 32KB, dual-ported data cache [1, 3].

The POWER4 chips were fabricated in the state-of-the-art IBM 0.18-μm CMOS 8S3 SOI (silicon-on-insulator) technology with seven levels of copper wiring [4]. Some of the features of this technology are given in **Table 1**. Characteristics of the POWER4 chip fabricated in this technology are given in **Table 2**. Using these chips, a 32-way SMP system has been operated in our laboratory at clock frequencies exceeding 1.3 GHz. Work is in progress to release the POWER4 design in CMOS 9S technology, which will significantly reduce the chip area as well as improve performance and decrease power dissipation.

The complexity and size of POWER4, together with its high operating frequency, presented a number of significant challenges for the design team. The chip complexity (as measured by transistor count) is five to ten times greater than that of chips available when the design of POWER4 was started, straining the capabilities of the design team, methodology, and tools to deliver a correctly functioning chip. The high-frequency goal for such a large chip placed stringent requirements on the circuits, clock network, and power distribution, as well as on the engineering of long wires in which signals required several clock cycles to traverse the chip in a frequently noisy environment. In addition, the design was carried out by a

0018-8646/02/$5.00 © 2002 IBM

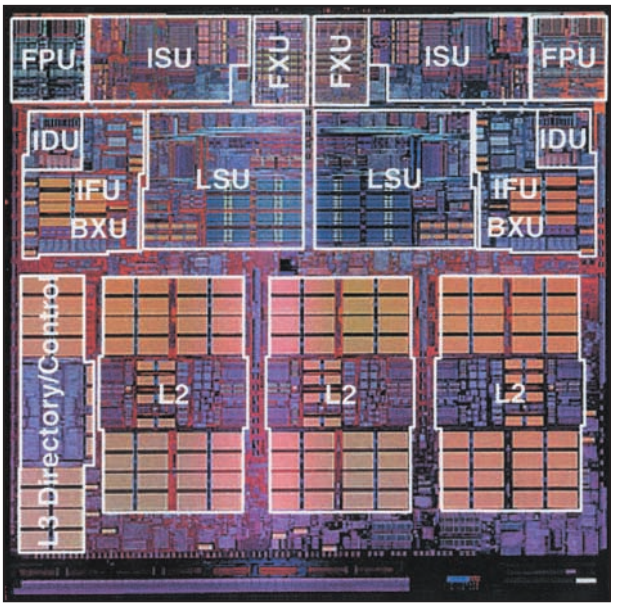

Figure 1

POWER4 chip photograph showing the principal functional units in the microprocessor core and in the memory subsystem.

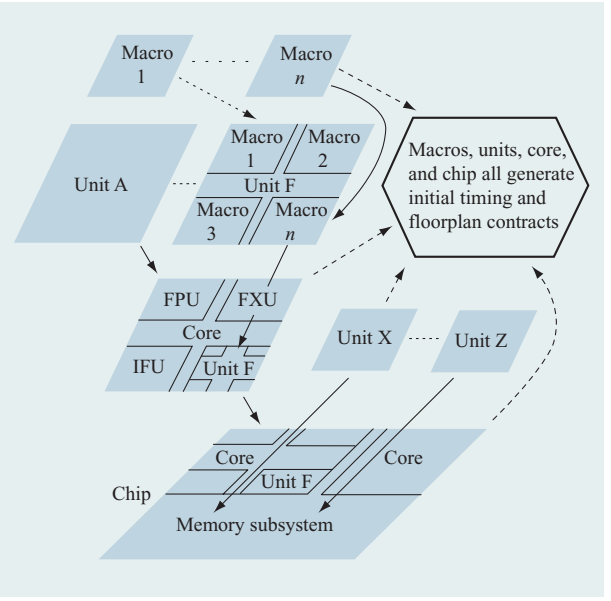

Figure 2

Elements in the physical and logical hierarchy used to design the POWER4 chip.

Table 1 Features of the IBM CMOS 8S3 SOI technology.

Gate L_{eff}	0.09 μm	
Gate oxide	2.3 nm	
Metal layers	pitch	thickness
M1	0.5 μm	0.31 μm
M2	0.63 μm	0.31 μm
M3–M5	0.63 μm	0.42 μm
M6 (MQ)	1.26 μm	0.92 μm
M7 (LM)	1.26 μm	0.92 μm
Dielectric ε_r	~4.2	
V_{dd}	1.6 V	

Table 2 Characteristics of the POWER4 chip fabricated in CMOS 8S3 SOI.

Clock frequency (f_c)	>1.3 GHz	
Power	115 W	(@ 1.1 GHz, 1.5 V)
Transistors	174,000,000	
Macros (unique/total)	1015	4341
Custom	442	2002
RLM	523	2158
SRAM	50	181
Total C4s	6380	
Signal I/Os	2200	
I/O bandwidth	>500 Mb/s	
Bus frequency	1/2 f_c	
Engineered wires	35K	
Buffers and inverters	100K	
Decoupling cap	300 nF	

multi-site team, necessitating the development of ways to synchronize the design environment and data (as well as the design team).

In the following sections of this paper, the design methodology, clock network, circuits, power distribution, integration, and timing approaches used to meet these challenges for the POWER4 chip are described, and results achieved for POWER4 are presented.

Design methodology

The design methodology for the POWER4 microprocessor featured a hierarchical approach across multiple aspects of the design. The chip was organized physically and logically in a four-level hierarchy, as illustrated in **Figure 2.** The smallest members of the hierarchy are "macros" typically containing 50 000 transistors. Units comprise approximately 50 related macros, with the microprocessor core made up of six units. The highest level is the chip, which contains two cores plus the units associated with the on-chip memory subsystem and interconnection fabric. This hierarchy facilitates concurrent design across all four levels. While the macros (blocks such as adders, SRAMs, and control logic) are being designed at the transistor and

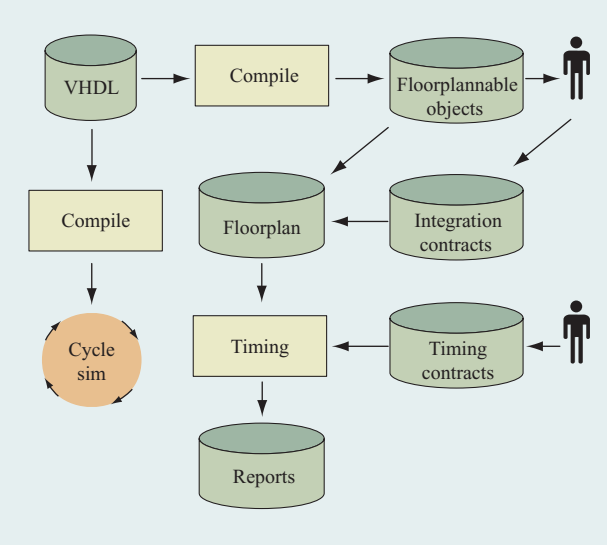

Figure 3

Design flow used during high-level design. The rectangular shapes represent tools used to complete a portion of the design. The cylinders represent design data, and the circle the verification of the VHDL logic.

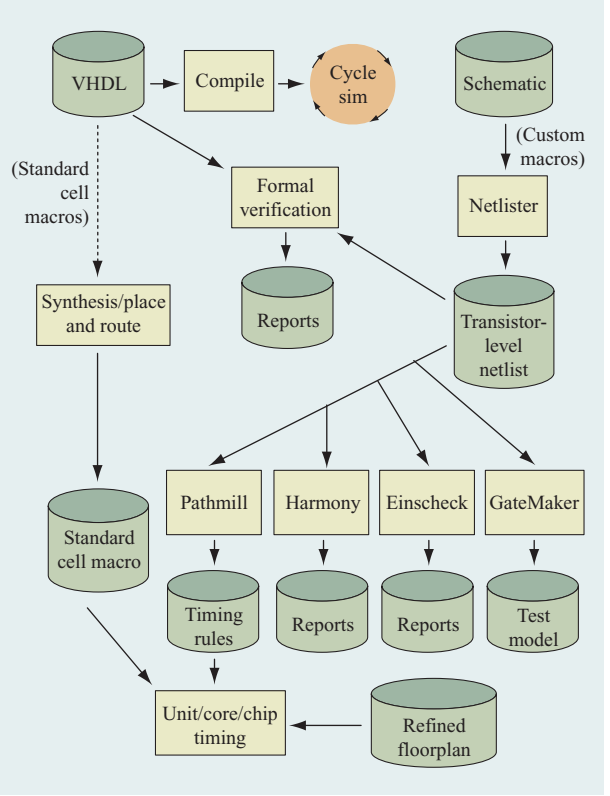

Figure 4

Design flow used during schematic design. The rectangular shapes represent tools used to complete a portion of the design. The cylinders represent design data and the circle the verification of the VHDL logic.

standard-cell level, the units and then in turn the core and chip are concurrently being floorplanned and timed. Provided that the proper contracts have been created, all of this work can proceed independently and in parallel. Further parallelism of design was employed by separating the design tasks of logic entry and simulation from those of circuit design, floorplanning, and timing. For each of the four hierarchies, the logic design and simulation were able to progress in parallel with the circuit design of that entity, with a final formal verification step to ensure equivalence [5].

Design phases

The design process was divided into several successive phases—high-level design, schematic design, and physical design—with increasing refinement of the design occurring at each phase. The design process flows for these phases are shown in **Figures 3–5**. At the start of high-level design, the chip was partitioned into chip, core, unit, and macro "blocks" as described above. The high-level logic is written in VHDL and compiled into physical blocks that match this hierarchy. Transistor-level and standard-cell-level design can then be performed in parallel with macro-level simulation and logic entry. Similarly, at the unit, core, and chip levels, floorplanning and even timing can begin in parallel with the higher-level logic design and simulation. The main deliveries from any block owner at this time are "contracts." Contracts are the early size and timing budgets that allow the design of various blocks, floorplanning, and timing to proceed in concert. At the completion of high-level design, consistent contracts are in place across all levels of the hierarchy.

The next phase, schematic design (Figure 4), is marked by complete transistor-level schematics for macros, including near-final transistor sizings along with R and C estimates for wires. These in turn allow more detailed floorplan abstracts of macros, complete with blockage maps for wiring on upper levels and sufficiently accurate timing rules. The newly refined floorplannable objects allow for concrete physical design at the unit, core, and chip levels, complete with engineered buses and known available routing tracks at all levels. Timing at this point is proceeding at all levels of the hierarchy, with the added degree of accuracy provided by the macro-level timing rules and R and C estimates for unit, core, and chip wires. During the schematic design phase, the logic design progresses toward completing the VHDL with ever-increasing confidence due to accumulating simulation

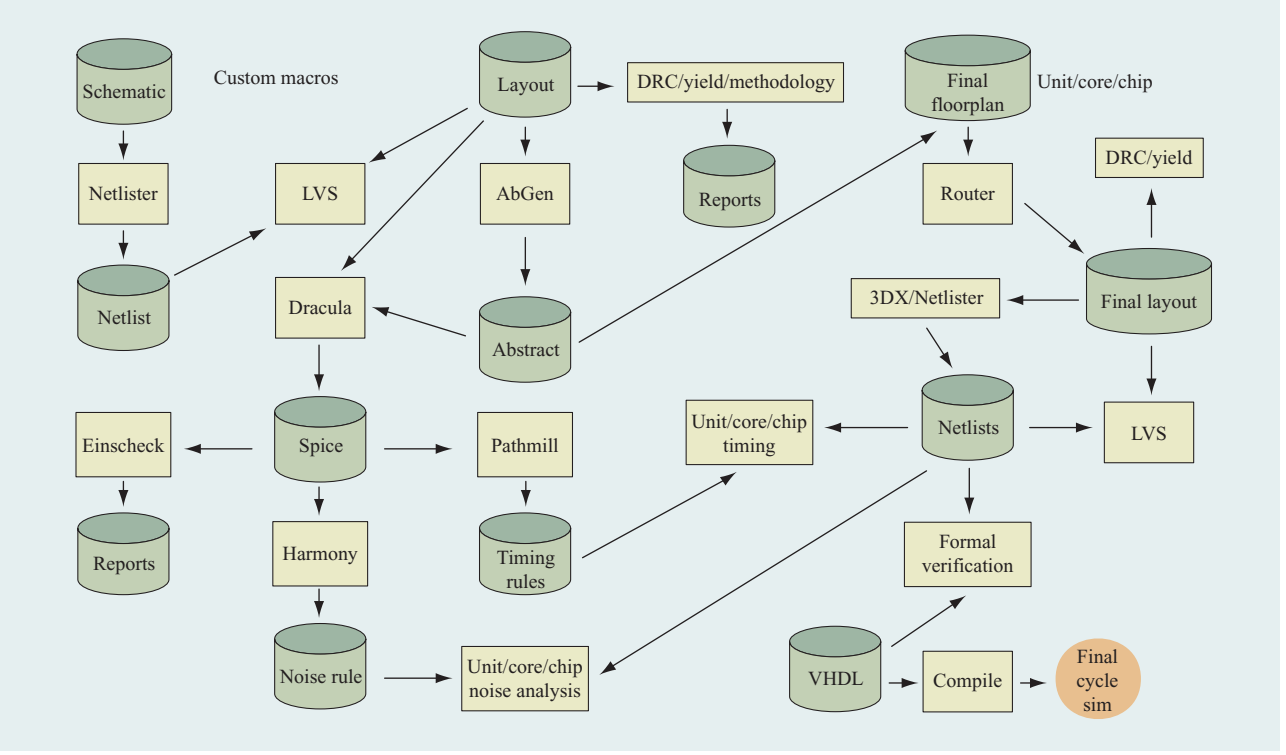

Figure 5

Design flow used during physical design. The rectangular shapes represent tools used to complete a portion of the design. The cylinders represent design data and the circle the verification of the VHDL logic.

at the unit, core, and chip levels. At this point, the schematics and VHDL are verified through formal verification at the macro level.

The final phase, physical design (Figure 5), begins when the logic has reached a high level of stability and when the timing goals have been met. It progresses until physical layouts exist for each macro, allowing the generation of final parasitic-extraction-based transistor-level timing rules, and detailed abstracts containing the macro sizes and complete metal blockage maps. The macros are folded into the unit-level integration and timing tasks as they are completed, replacing schematic-based timing rules and abstracts. Final unit wiring and extraction, final core wiring and extraction, and final chip wiring and extraction are performed at this time, leading to the final timing runs. The now completely simulated logic is verified against the physical design, and the chip design is complete.

Design flow and tools

At the start of the POWER4 microprocessor design, a tool suite and methodology were put together by picking the best elements of established IBM microprocessor design methodologies, such as POWER3, S/390* G4, and PowerPC 615 [6, 7], and combining them with new ideas specific to this design. A mix of IBM-developed and vendor-supplied design automation software was used to provide a transistor-level design system at the macro level and a quick-turnaround, convergent design system at the upper levels of the hierarchy. The tools and methodology could be employed on the same design across six IBM sites: Austin, Yorktown, Poughkeepsie, Burlington, Rochester, and Fishkill. This was accomplished in large part due to the Common Tools Environment (CTE) [8]. The 174 million transistors that make up the chip are partitioned into more than 1000 unique macros, many of which were used several times. To ensure that each of these macros was production-ready, an elaborate design data audit system was employed [9]. Each macro was graded (A, B, C, D, F) on the basis of the results of running each of 26 tools. This resulted in more than 16000 grades which had to be an A before the chip was released to manufacturing. Roughly 500 of the macros were synthesized standard-cell macros, or RLMs (random logic macros) built using automated placement and routing. Here we were able to combine IBM BooleDozer* [10] and

placement and wiring by Cadence** to create a physically aware synthesis system which resulted in meaningful first-pass standard-cell macro timings and physical designs.

The remaining macros were primarily custom macros that were designed and verified using a transistor-level methodology. At the start of the design, a design guide was written with several chapters containing explicit rules governing the design of the custom macros. An electrical circuit-checking tool was developed (EinsCheck) which verified that these rules were met. The scope of the rules spanned transistor topology, transistor sizings, loading, and *RC* delays. The custom macros were timed at the transistor level using the vendor-supplied tool Pathmill.** By timing at the transistor level, inaccuracies and needless pessimism that arise from a block-based timing approach were avoided. Noise analysis was also performed at the transistor level for the custom macros using the IBM EDA-developed tool Harmony [11]. For both timing and noise analysis, each run on a custom macro served two purposes. First, a detailed report was provided which showed the timing and noise characteristics of the macro internals; second, timing and noise rules were produced which were used for timing and noise analysis at upper levels of the hierarchy. All three of the transistor-level tools mentioned above—EinsCheck, Pathmill, and Harmony—were used both in the schematic design phase, in which transistor parasitics and wiring resistance and capacitance were estimated, and in the physical design phase, in which each custom macro was extracted using the Dracula** tool. Formal verification was also performed at the transistor level by creating a netlist from the schematic and proving equivalence with the VHDL, using the IBM tool Verity [12]. Analogously, test-model generation was performed by IBM GateMaker [13], which used the transistor-level netlist to create the test model. Finally, each macro underwent physical verification by running design-rule checking (verifying that all shapes-based ground rules had been observed), logical-to-physical verification (ensuring that the schematic and the layout were equivalent), methodology checks (special shapes-based checking which ensured that the macros could be assembled into a unit), and yield checking (a shapes-based check which looked for potential yield detractors). These physical verification checks were implemented in an IBM EDA-developed tool, Niagara. In addition, one extra set of methodology checks were performed on each macro using code written in the Cadence SKILL** language.

Clock design

A high-quality global clock signal must be distributed to every latch and clocked circuit for the success of a high-frequency microprocessor. The global clock distribution becomes especially challenging for such a large and complex chip because of the longer wires and gain needed to drive the large distributed clock load. An SOI-specific phase-locked-loop (PLL) design and a simple yet extensively optimized clock-distribution strategy achieved an unexcelled measured jitter of 35 ps and skew of 20 ps. Simple clock timing and rapid bring-up were facilitated by using a single chip-wide global clock domain with no skew feedback or adjustment settings.

A single PLL was used, placed near the center of the chip to minimize the global clock-distribution delay. An analog power supply was generated on chip for the analog PLL circuits using a capacitor mounted on the surface of the chip module. The PLL oscillator runs at twice the chip clock frequency and is divided by 2 to generate a 50% duty-cycle global clock.

The design of the PLL was SOI-specific, adapted from a previous design [14]. The primary focus of the SOI design was to add body contacts to the analog circuits while minimizing the body resistance and the device-width uncertainty that accompany a body-contacted device. In addition, the capacitor structures used in the PLL were modified to maintain leakage levels acceptable for correct circuit operation. The SOI capacitor structures were improved by adding a thicker oxide structure and through strategic layout of the capacitor recessed-oxide openings.

The SOI PLL design resulted in less than 10 ps of cycle compression from the PLL and 35 ps total compression from the PLL plus clock distribution. The chip-to-chip jitter, defined as the maximum phase difference between clock edges on different processor chips, is 150 ps. The primary goals of minimum clock uncertainty margin and timing simplicity were achieved through the design and tuning of the single global net that covered the entire chip. Since the resulting modeled skew was negligible, the clock timing consisted only of an uncertainty due to model and process errors. For simplicity, this was assumed to be identical for each of the 15 200 global clock pins. The ability to neglect modeled skew and use a simple constant clock uncertainty across the whole chip contributed to rapid chip-timing progress.

Global clock-distribution network

The global clock-distribution and tuning strategy used was an extension of previous server microprocessor designs [15]. The topology is shown schematically in **Figure 6**. **Figure 7** illustrates this network in greater detail using a 3D visualization showing all wire and buffer delays. The first part of the clock distribution consists of buffered H-trees, designed as symmetrically as possible, which drive the final set of 64 carefully placed sector buffers (shown in Figure 7). The 64 sector buffers each drive a tunable sector tree network, designed for minimum delay without length matching. These final sector trees all drive a single full-chip clock grid (appearing at the top of Figure 7) at 1024 evenly spaced points. We also see from Figure 7 that

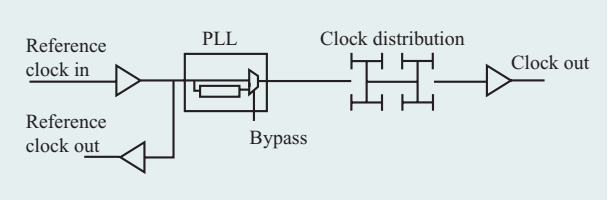

Figure 6

Schematic diagram of global clock generation and distribution.

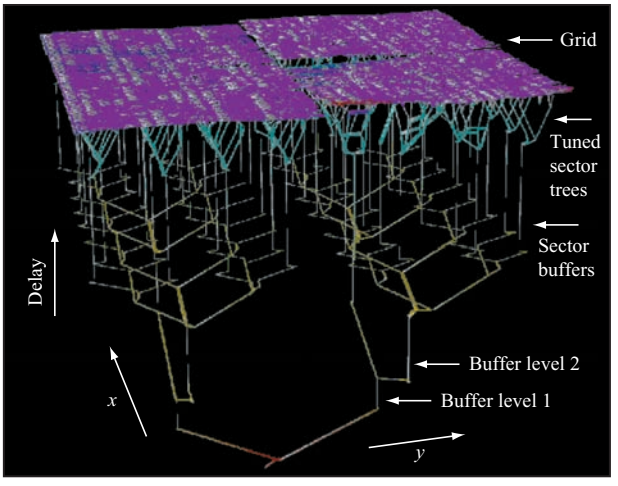

Figure 7

3D visualization of the entire global clock network. The x and y coordinates are chip x, y, while the z axis is used to represent delay, so the lowest point corresponds to the beginning of the clock distribution and the final clock grid is at the top. Widths are proportional to tuned wire width, and the three levels of buffers appear as vertical lines.

the total wire delay is similar to the total buffer delay. A patented tuning algorithm [16] was required to tune the more than 2000 tunable transmission lines in these sector trees to achieve low skew, visualized as the flatness of the grid in the 3D visualizations. **Figure 8** visualizes four of the 64 sector trees containing about 125 tuned wires driving 1/16th of the clock grid. While symmetric H-trees were desired, silicon and wiring blockages often forced more complex tree structures, as shown. Figure 8 also shows how the longer wires are split into multiple-fingered transmission lines interspersed with V_{dd} and ground shields (not shown) for better inductance control [17, 18]. This strategy of tunable trees driving a single grid results in low skew among any of the 15 200 clock pins on the chip, regardless of proximity.

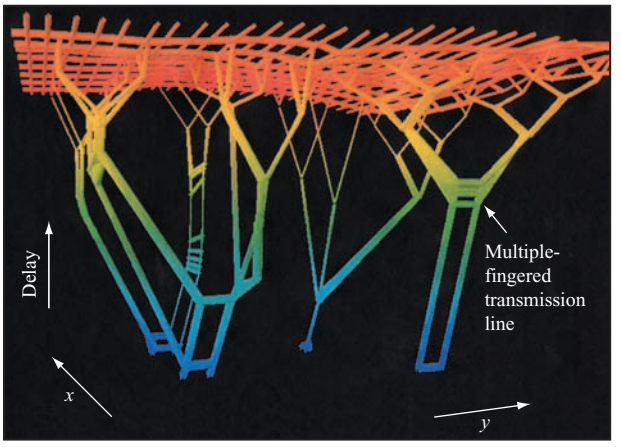

Figure 8

Visualization of four of the 64 sector trees driving the clock grid, using the same representation as Figure 7. The complex sector trees and multiple-fingered transmission lines used for inductance control are visible at this scale.

From the global clock grid, a hierarchy of short clock routes completed the connection from the grid down to the individual local clock buffer inputs in the macros. These clock routing segments included wires at the macro level from the macro clock pins to the input of the local clock buffer, wires at the unit level from the macro clock pins to the unit clock pins, and wires at the chip level from the unit clock pins to the clock grid.

Design methodology and results

This clock-distribution design method allows a highly productive combination of top-down and bottom-up design perspectives, proceeding in parallel and meeting at the single clock grid, which is designed very early. The trees driving the grid are designed top-down, with the maximum wire widths contracted for them. Once the contract for the grid had been determined, designers were insulated from changes to the grid, allowing necessary adjustments to the grid to be made for minimizing clock skew even at a very late stage in the design process. The macro, unit, and chip clock wiring proceeded bottom-up, with point tools at each hierarchical level (e.g., macro, unit, core, and chip) using contracted wiring to form each segment of the total clock wiring. At the macro level, short clock routes connected the macro clock pins to the local clock buffers. These wires were kept very short, and duplication of existing higher-level clock routes was avoided by allowing the use of multiple clock pins. At the unit level, clock routing was handled by a special tool, which connected the macro pins to unit-level pins, placed as needed in pre-assigned wiring tracks. The final connection to the fixed

clock grid was completed with a tool run at the chip level, connecting unit-level pins to the grid. At this point, the clock tuning and the bottom-up clock routing process still have a great deal of flexibility to respond rapidly to even late changes. Repeated practice routing and tuning were performed by a small, focused global clock team as the clock pins and buffer placements evolved to guarantee feasibility and speed the design process.

Measurements of jitter and skew can be carried out using the I/Os on the chip. In addition, approximately 100 top-metal probe pads were included for direct probing of the global clock grid and buffers. Results on actual POWER4 microprocessor chips show long-distance skews ranging from 20 ps to 40 ps (cf. **Figure 9**). This is improved from early test-chip hardware, which showed as much as 70 ps skew from across-chip channel-length variations [19]. Detailed waveforms at the input and output of each global clock buffer were also measured and compared with simulation to verify the specialized modeling used to design the clock grid. Good agreement was found. Thus, we have achieved a "correct-by-design" clock-distribution methodology. It is based on our design experience and measurements from a series of increasingly fast, complex server microprocessors. This method results in a high-quality global clock without having to use feedback or adjustment circuitry to control skews.

Circuit design

The cycle-time target for the processor was set early in the project and played a fundamental role in defining the pipeline structure and shaping all aspects of the circuit design as implementation proceeded. Early on, critical timing paths through the processor were simulated in detail in order to verify the feasibility of the design point and to help structure the pipeline for maximum performance. Based on this early work, the goal for the rest of the circuit design was to match the performance set during these early studies, with custom design techniques for most of the dataflow macros and logic synthesis for most of the control logic—an approach similar to that used previously [20]. Special circuit-analysis and modeling techniques were used throughout the design in order to allow full exploitation of all of the benefits of the IBM advanced SOI technology.

The sheer size of the chip, its complexity, and the number of transistors placed some important constraints on the design which could not be ignored in the push to meet the aggressive cycle-time target on schedule. These constraints led to the adoption of a primarily static-circuit design strategy, with dynamic circuits used only sparingly in SRAMs and other critical regions of the processor core. Power dissipation was a significant concern, and it was a key factor in the decision to adopt a predominantly static-circuit design approach. In addition, the SOI technology,

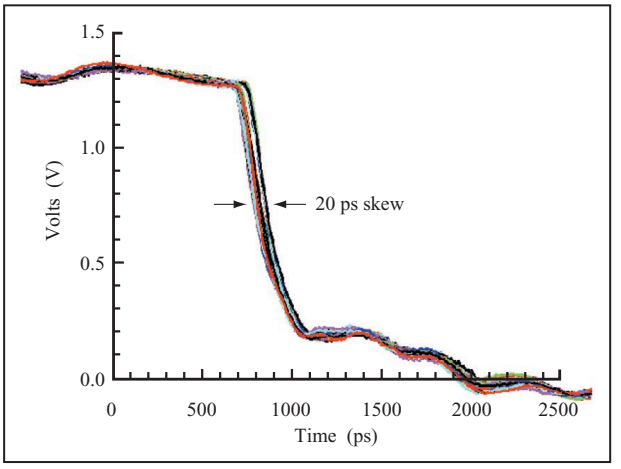

Figure 9

Global clock waveforms showing 20 ps of measured skew.

including uncertainties associated with the modeling of the floating-body effect [21–23] and its impact on noise immunity [22, 24–27] and overall chip decoupling capacitance requirements [26], was another factor behind the choice of a primarily static design style. Finally, the size and logical complexity of the chip posed risks to meeting the schedule; choosing a simple, robust circuit style helped to minimize overall risk to the project schedule with most efficient use of CAD tool and design resources. The size and complexity of the chip also required rigorous testability guidelines, requiring almost all cycle boundary latches to be LSSD-compatible for maximum dc and ac test coverage.

Another important circuit design constraint was the limit placed on signal slew rates. A global slew rate limit equal to one third of the cycle time was set and enforced for all signals (local and global) across the whole chip. The goal was to ensure a robust design, minimizing the effects of coupled noise on chip timing and also minimizing the effects of wiring-process variability on overall path delay. Nets with poor slew also were found to be more sensitive to device process variations and modeling uncertainties, even where long wires and *RC* delays were not significant factors. The general philosophy was that chip cycle-time goals also had to include the slew-limit targets; it was understood from the beginning that the real hardware would function at the desired cycle time only if the slew-limit targets were also met.

The following sections describe how these design constraints were met without sacrificing cycle time. The latch design is described first, including a description of the local clocking scheme and clock controls. Then the circuit design styles are discussed, including a description

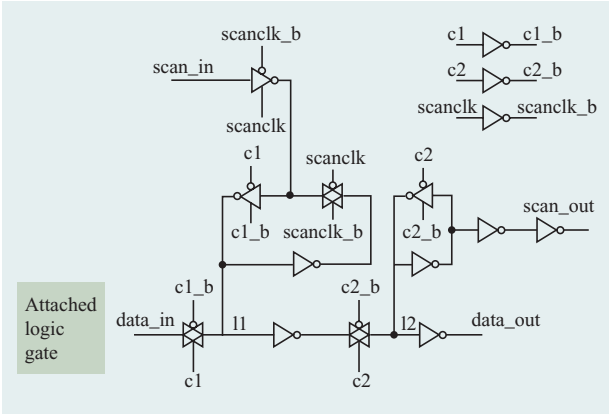

Figure 10

Standard transmission-gate master–slave flip-flop with LSSD capability. The input logic gate is specified independently by the designer, with constraints on gate type, drive strength, and physical proximity.

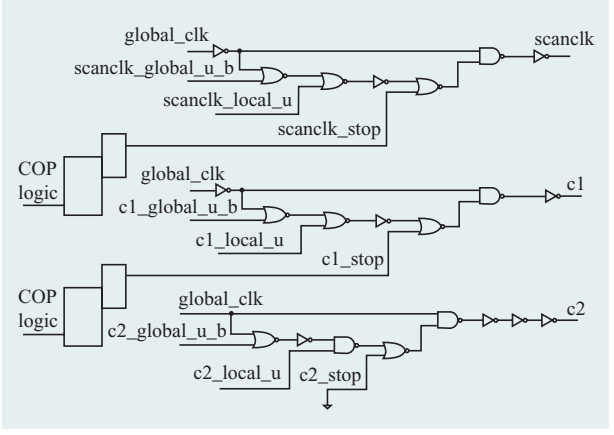

Figure 11

Local clock buffers for master–slave latch, including pipeline latches for ac test control signals. The pin "local_u" can be tied either to a supply ("nominal" case) or to the global clock (rising edge delayed) on a buffer-by-buffer basis. Pipeline latches for COP control signals were usually shared among a number of local clock buffers.

of some of the special techniques used to enhance performance. Finally, the implementation of both custom dataflow designs and control RLMs is discussed.

Latch design

By far the majority of the latches in the design were conventional transmission-gate master–slave flip-flops, as shown in **Figure 10**. These latches were designed to minimize exposure to noise-induced upset, to provide a low soft-error-rate (SER) exposure, and to be generally tolerant of a certain amount of local clock skew. In general, designers were given some degrees of freedom in specifying the latch parameters, and the final designs were subject to a layout-based checking routine to verify compliance with all latch, clocking, and other circuit design rules. As shown in Figure 10, designers were allowed to customize the logic gate which drives into the first transmission gate, thereby minimizing the overhead imposed by the latch. The transmission gates and the two inverters in the data path could also be sized by designers (within specified limits) in order to separately control and optimize the latch power, setup time, and clock-to-data-out delay.

The two local clock phases (c1 and c2), as well as the scan clock, were derived locally from one tap of the global clock, as shown in **Figure 11**. Each local clock buffer contained two control inputs for test and debug capability. In addition, where extra margin was needed for protection against race conditions, the designer could make use of the "local_u" setting as shown, in order to delay the rising clock edge on a buffer-by-buffer basis. The "global_u" signals were sourced from scan-only latch banks inside the functional units, and were used to selectively delay the rising clock edge for debugging purposes. Each global signal was routed to a large number of local clock buffers, with a partitioning strategy determined by the circuit and logic teams. The stop controls were pipelined from the chip onboard processor (COP) to all clock buffers across the chip, with the final segment from the local pipeline latch into the buffer having to occur in half a clock cycle. The c1 and scan clock buffers had separate stop controls, allowing arbitrary sequencing of the scan and c1 (system) clocks. For most of the latches, the c2 clock was free-running, with the stop signal tied to ground. However, outside the processor core, certain regions of logic were designed to operate at integer multiples of the core cycle time. In this case, pipeline stop signals were used to fire both c1 and c2 clocks in a programmable fashion, thereby setting the frequency for logic operation in a given domain. For example, firing the clocks on alternate cycles would be appropriate for a 2:1 frequency reduction, every third cycle for 3:1 operation, etc.

For critical timing paths, designers were given the means to reduce the overhead imposed by the standard flip-flop, which included two embedded inverters and a built-in setup time penalty for potential clock skew, process variability, and across-chip linewidth variation (ACLV). Separate master and slave latch designs were provided, which could be inserted at arbitrary points in the logic. As shown in **Figure 12**, this allows logic signals on critical timing paths to propagate through alternating

cycle-boundary (master, or c1) and mid-cycle (slave, or c2) latches without incurring a setup time penalty. Latch placement is also simplified. On average, a half cycle of logic is allowed between c1 and c2 latches or between c2 and c1 latches. However, less logic between any two latches means that time is given up to the logic following the receiving latch, and more logic means that time is taken from the following logic. **Figure 13** shows an example of a master-only latch with LSSD compatibility [2]. The area overhead for LSSD compatibility is significant in this case, since an additional c2 latch must be provided (aside from the separate c2 added to the downstream logic) for scan functionality. However, even in this situation, the extra area was a relatively small addition to the overall total, and the flexibility of this scheme would often allow area savings in other parts of the design.

Although the split-latch scheme offered many benefits in terms of its flexibility, resistance to clock skew, process variation, and overall reduction of the latch overhead on critical paths, there were some substantial drawbacks to the use of these latches in the processor design. Most notable among these was the increased difficulty of timing paths through logic containing these latches. The timing tool had to be able to deal with multi-cycle paths through transparent latches, including loops and other difficult topological situations, and then had to present the timing data in an intelligible way. In addition, there were certain ac test issues which had to be addressed, including the fact that it became difficult to assess how many back-to-back cycles would be needed to capture all of the critical timing paths through the machine. Also, ac timing failures could become much more difficult to debug, at least in principle.

Circuit styles

For reasons mentioned earlier, complementary static circuits were used predominantly in the design. The designers were allowed to use these circuits with few restrictions. Limits were placed on effective beta ratio (p-to-n strength ratio), minimum device size (for tracking and process variability concerns), maximum stack height, and node slew (for noise, tracking, and process/model sensitivity reasons). Although the maximum allowed stack heights were 4 and 3 for n-FETs and p-FETs, respectively, the recommended procedure was to limit designs (especially in timing-critical paths) to 3 and 2 for n-FETs and p-FETs, respectively. This recommendation was made on the basis of the sensitivity of wide gates to simultaneously switching inputs, model uncertainties in the linear regime of the device operation, sensitivity to body voltage, and the desire to avoid having any one gate or only a few gates contribute a large fraction of the delay in a given cycle.

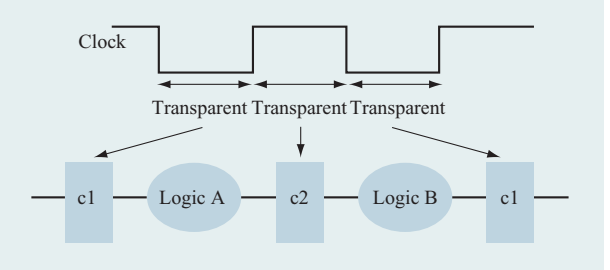

Figure 12

Split-latch clocking diagram. c1 and c2 latches are transparent on alternate half-cycles, allowing logic to propagate through successive cycles of logic in a way that is tolerant of any local clock skew. In addition, variability in timing delays due to modeling uncertainty or process linewidth variation can be averaged out over several cycles, since time can be "borrowed" from the following cycle if the signal arrives a little late at a given latch, or "given" to the following cycle if the signal is early.

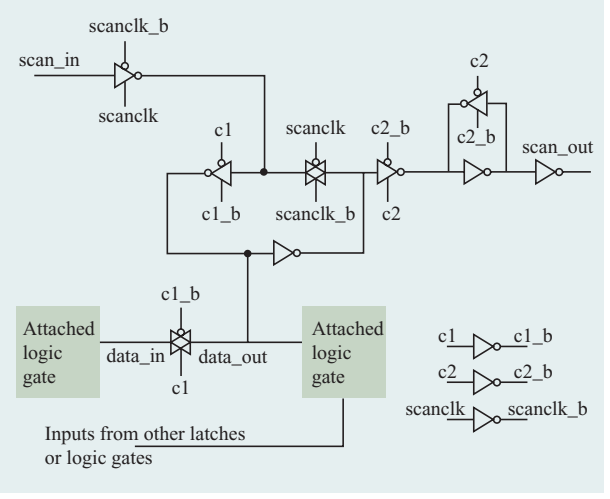

Figure 13

Scannable split-latch (c1) design. Designers were allowed to tune the transmission gate size, and could specify input and output gates (with constraints on gate type, drive strength, output load, and physical proximity).

The technology also offered reduced-V_t (threshold voltage) n-FET and p-FET devices, which provided a performance improvement of about 10% over the standard V_t counterparts, but at the cost of higher off-state leakage currents. Therefore, it was desirable to use these low-V_t devices selectively, in only the most timing-critical parts of the design. The process technology was planned such that the ground rules allowed swapping from normal V_t to low V_t at the last moment, usually with minimal or no other

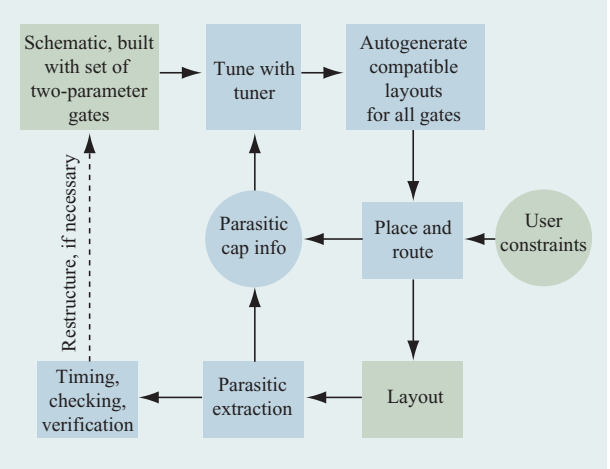

Figure 14

Rapid-turnaround-time semicustom design process. Design automation at each step allowed rapid iteration and optimization of the design, and also provided a means of exploring a much wider design space than would otherwise have been possible.

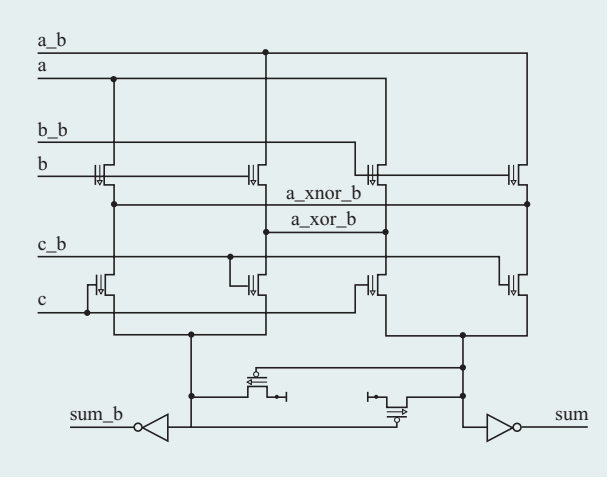

Figure 15

Circuit schematic of part of the 3:2 compressor used in the floating-point multiplier array, implemented in CPL logic.

effect to the circuit layout. In this way, the decision to use low-V_t devices could be postponed until near the end of the design process, and low-V_t devices could be selectively employed only where necessary. Overall, less than 10% of the logic transistors were implemented with low-V_t devices, thereby avoiding a large increase in chip standby current while still capturing the performance benefit of these devices.

One particular benefit of using static circuits in the design was that this design style allowed designers to make full use of various design automation aids, including an advanced circuit tuner [28, 29] providing rapid design turnaround for investigation of different circuit topologies and implementations, and detailed performance tuning based on layout parasitics. An example of how this process works is shown in **Figure 14** [30]. In the instruction-fetch/branch unit, these techniques were used to build a series of eight compact static carry-lookahead Ling adders [31] used for branch-target address calculation. The delay through the 24-bit adder and built-in four-way multiplexor was reduced to about 400 ps (or a little over 9 FO4, where one FO4 delay unit is the average delay of an inverter with a fan-out of 4). This adder ended up being nearly as fast as the dynamic adder originally considered for this function, but at a fraction of the cost in terms of power, and with improved allowance for overhead routability.

In addition to complementary static circuits, an attempt was made to use certain circuit styles thought to be more optimal for use in SOI technology. One such family used was complementary passgate logic (CPL), an example of which is shown in **Figure 15**. In these circuits, coupling between source/drain and the body of the passgate device helps to raise the body voltage during a $0 \rightarrow 1$ transition, lowering the threshold voltage and speeding the transition. However, such circuits tend to operate in regions where the device models are less accurate, they are sensitive to variations in threshold voltage (especially in the linear regime), and they are also typically very sensitive to history-dependent delay effects [32]. These issues meant that substantial extra design timing margin had to be added to ensure that these circuits met the chip cycle-time design targets.

Another circuit style used rather widely involved networks of transmission-gate circuits, an example of which is shown in **Figure 16** [error check and correction (ECC) logic in the L2 control]. Since the ECC function is on the critical path for accessing the L2 cache upon an L1 miss, the Hamming matrix was optimized for the lowest number of logic levels and highest number of shared terms without compromising the single-error correction–double-error detection (SEC–DED) requirement. The X(N)OR4 gate was the best choice as a compact fast building block for this function, since the H-matrix could be optimized to map directly into such a gate representation with a minimum number of terms. XOR logic functions are particularly well suited for implementation with passgate networks, and the addition of the complementary p-FET in each transmission gate, along with careful gate tuning and slew optimization, ensured that the design had the required degree of robustness and insensitivity to process variations. The use of low-V_t gates was also very effective

in improving the speed and slew characteristics of these circuits.

Finally, a number of dynamic circuits were used on the chip for certain critical applications. The usage of dynamic circuits was limited to some of the register files, array circuitry, and some circuitry in the load/store unit, designed to interface smoothly with the dynamic circuitry in the data cache array. Both footed and non-footed domino circuits were used, with special techniques used to overcome unique SOI noise issues [24, 25] and a variety of tools to check for noise exposures or other weaknesses in the design. In addition, any half-cycle timing paths were designed with extra margin to account for possible duty-cycle variations in the clock waveforms.

Circuit design and layout

The most critical designs employed full-custom techniques for both the circuit design and the subsequent layout. Critical design aspects which drove this level of customization could include timing/cycle-time issues, area issues, and/or routability/wiring issues. These designs (usually dataflow structures), typically took the longest amount of time to implement, but offered the most performance potential, with generally more optimal usage of area and wire resources. Often these designs contained many reusable components (multiplexors, latch cells, clock buffers, etc.) in order to reduce the resources required for implementation. In addition, larger blocks inside custom macros (adders, incrementors, comparators) were often built using the semicustom techniques described in the previous section. Area, power, pin placement, and timing of these macro sub-blocks could be customized depending on the application. A typical custom macro partition might contain from 20 000 to 50 000 transistors. Overall, the chip contained more than 400 unique custom macro blocks.

"Bit stacking" of standard-cell library "books" was another technique used successfully for implementation of certain large dataflow structures in regions of the chip where area and wiring channels were at a premium. As the name suggests, large structures were organized in a dataflow-like manner, with special techniques used to organize and place the individual library books. This allowed fast implementation of complex dataflow structures, with considerable ability to make changes late in the design cycle, while still maintaining the dataflow structure to minimize usage of wiring resources and to organize the flow of signals/data through the logic.

Control logic and standard-cell library

The control logic was implemented in the form of discrete RLMs, implemented with gates from a standard-cell library [33]. The control logic was synthesized from the high-level VHDL description, with various degrees of customization and optimization options

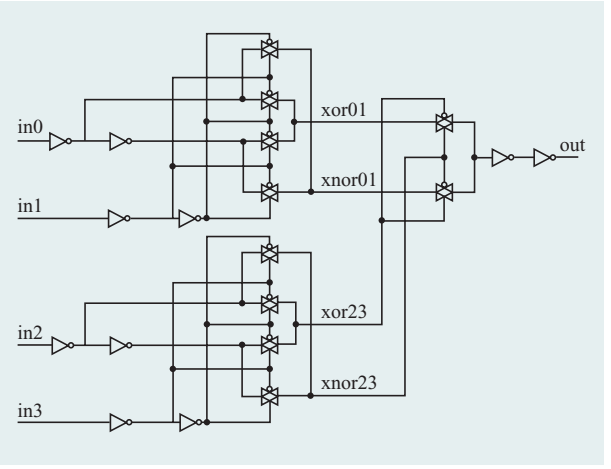

Figure 16

An xor4 gate used in the error check and correction (ECC) logic in the L2 control, implemented with low-V_T gates. The ECC logic function was well suited to implementation with transmission gate circuits.

available to improve the initial result from synthesis. After synthesis, a variety of specialized place-and-route techniques were used to implement the design, with iterations back through an incremental synthesis process to optimize the clock network, tune critical timing paths through the design, and fix node slew violations. In some cases, the resulting design was extracted and timed using the same methodology as that used for the custom macros, although in many cases the RLM was timed at the gate level, using the library timing rules. In either case, this automated design flow allowed control logic debug and timing work to continue to a very late stage in the design process without gating the schedule of the final design tape-out. This allowed the control logic to be implemented in such a way that it did not limit the overall chip cycle time.

The structure and the various special features of the standard-cell library were crucial to the success of the effort to close timing on the control logic. The library consisted of a relatively narrow series of simple logic gates, but each logic gate was supported with a wide matrix of options including a broad range of device widths, tapered stacks, several n:p width ratios, and normal- vs. low-device-threshold-voltage (V_t) books (**Table 3**). The goal was to allow the designer to achieve a result which was as close to a full-custom result as possible. To this end, a number of special latch cells were built with an integrated logic book (**Figure 17**), which allowed merging logic with the latch cell but avoided exposure of the latch transmission gate input to potentially noisy wires. Split-latch designs were also

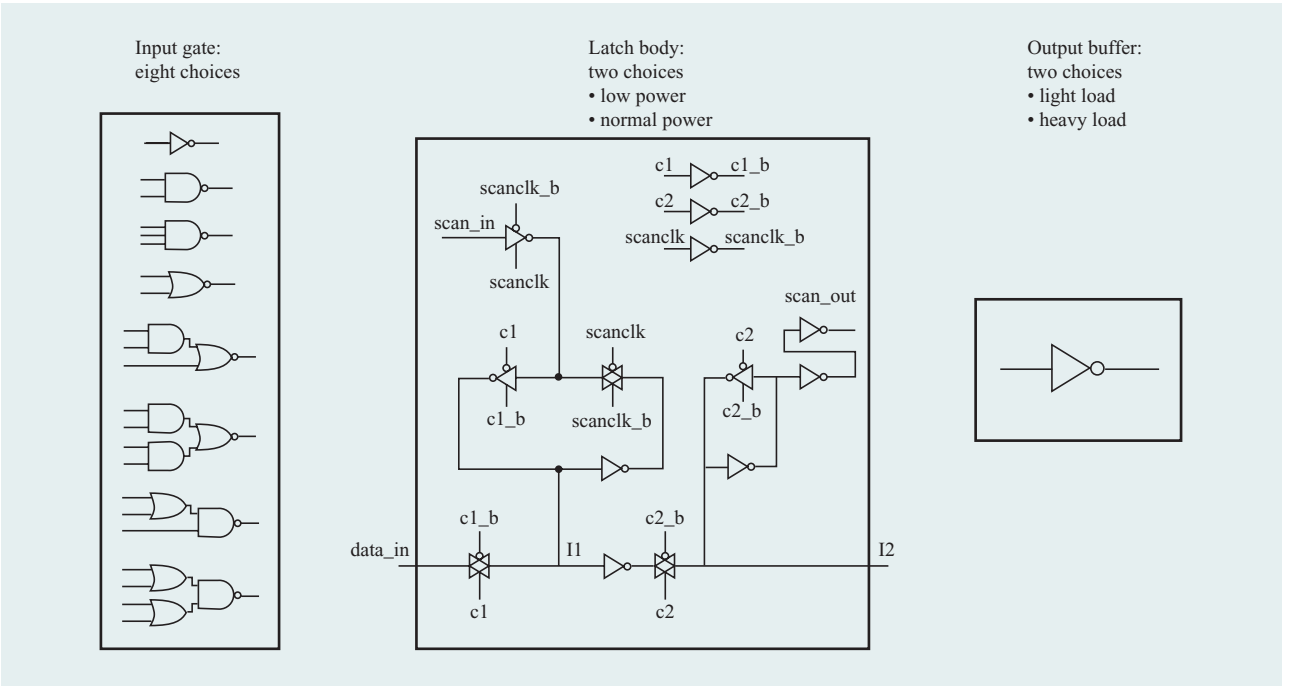

Figure 17

Library master–slave flip-flop with integrated front-end logic gate choices.

Table 3 Standard-cell library composition (excluding latches and clock buffers), showing the width and depth and some of the special features available. Limited numbers of special-purpose cells, including XOR/XNOR gates, were also included in the library.

Logic gate	*Beta ratios available*	*Power levels (input cap)*	*Number of V_t options*	*Number of tapered cells*	*Total number of cells*
INV	5	34	2	0	340
NAND2	5	29	2	28	318
NAND3	4	23	2	22	206
NAND4	3	16	2	0	96
NOR2	5	16	2	16	176
NOR3	3	6	2	0	36
AOI21	5	16	2	0	160
AOI12	5	16	2	0	160
AOI22	5	16	2	0	160
OAI21	5	16	2	0	160
OAI12	5	16	2	0	160
OAI22	5	16	2	0	160

available, with an integrated front-and-back logic gate (**Figure 18**). These designs gave the RLM designer almost the same ability as the custom designer to minimize the latch overhead and provide for clock-skew-tolerant operation.

The library also supported a number of optimization procedures which could be carried out after all other physical design steps were completed, without requiring any placement or wiring changes. On critical paths, the final, layout-based macro timing could be improved by automatically replacing standard-V_t books with their low-V_t counterparts, which were designed to have completely compatible footprints. This technique ensured that such low-V_t books (with higher performance but also higher leakage) were used only where necessary. In addition, all latch books which were not driving critical signals could be

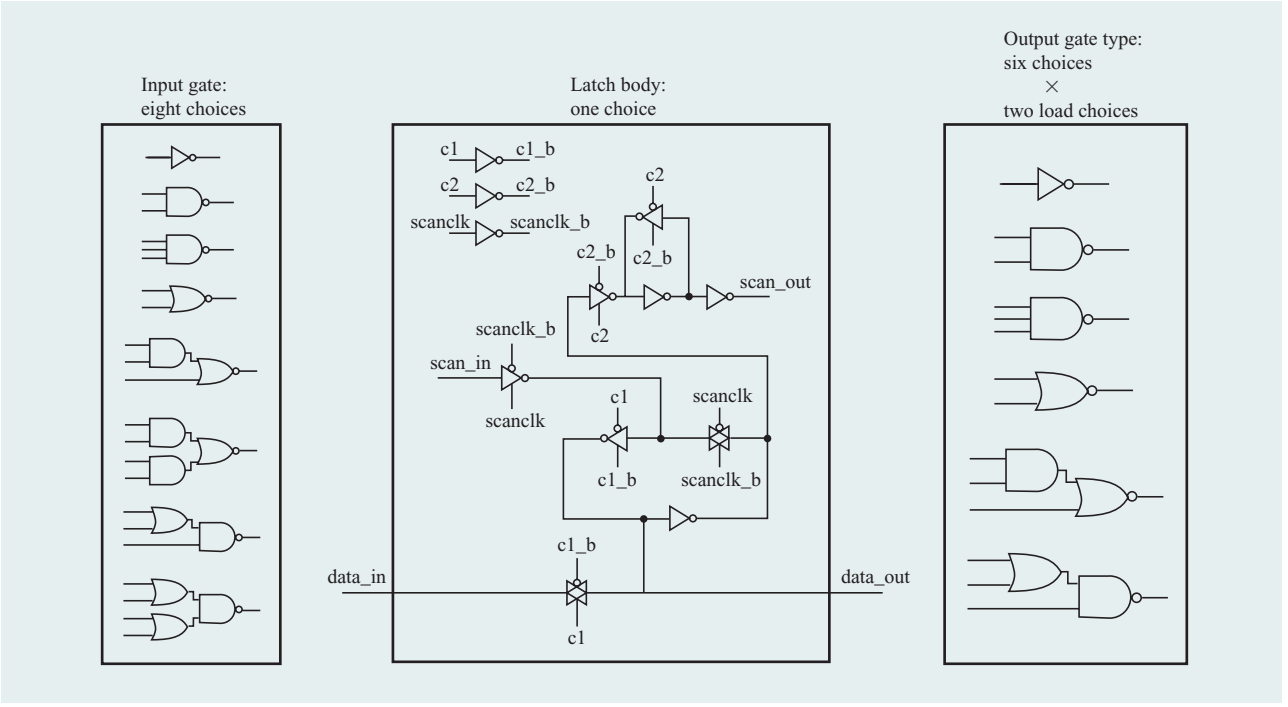

Figure 18

Split-latch design for standard cell library, showing front- and back-end logic gate choices. For the output gate, the latch drives one input; other inputs come from other logic blocks (same clock phase as latch output).

replaced with low-power books (along with the local clock buffer), again with footprint-compatible cells. Since a large fraction of the chip power was devoted to clocking and latching, this technique was able to offer significant power savings in the RLM macros. Finally, all white space inside the RLMs was filled with a combination of special decoupling capacitor cells and a gate-array backfill. The gate-array backfill could be customized into logic gates (inv, nand2, nand3, nor2) using only the metal design layers, allowing logic fixes to be made with changes to only the metal layers. This quick-fix capability allowed for faster turnaround of late logic fixes, significantly speeding up the overall system bringup time.

Power distribution

The design of the power-distribution network had three principal constraints based on our previous experience. First, the average dc voltage drop for the chip had to be limited to less than 30 mV for power densities up to 1 W/mm^2. Second, transient power fluctuations occur when the total chip power changes abruptly over a few cycles, causing the power supply to oscillate at the resonant frequency of the package/chip; the power network design had to limit such fluctuations to be less than ±10% of the nominal power supply. And finally, the common-mode noise on global interconnects had to be less than 200 mV for the worst-case wide-bus activity.

Because similar objectives had been imposed on earlier IBM microprocessor designs, the methods for analyzing and accomplishing the first two targets were well understood: A dc circuit analysis of the proposed power grid in regions with sparse power and ground C4s would predict worst-case dc drop with sufficient accuracy. A series of transient analyses of the chip/package during maximum power change would predict how much decoupling capacitance was required to hold fluctuations to 10% of supply voltage. For a total POWER4 chip power of ~115 W and a maximum transient power change of ~25 W, it was necessary to embed 250 nF of decoupling capacitance into critical areas of the chip.

These values of the total chip power and the maximum transient power change were determined by combining circuit power analysis results with unit-level logic simulation results. The circuit power analysis yielded power-dissipation equations as functions of input switching factors, while the logic simulation analysis yielded worst-case switching factors on a unit-by-unit basis.

The circuit power equations were obtained by analyzing every circuit and macro schematic with CPAM [34].

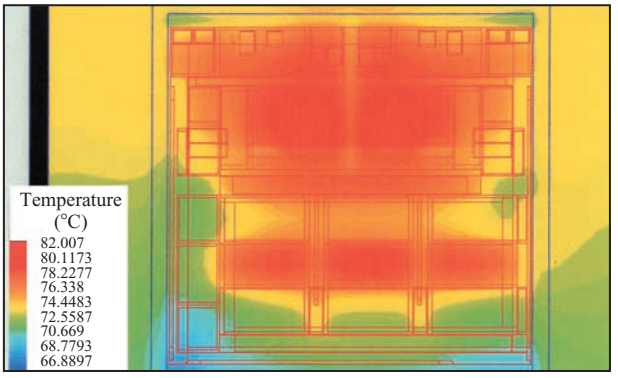

Figure 19

Map of FET junction temperatures for a 115-W packaged POWER4 chip derived from the chip power analysis and thermal modeling simulations described in the section on distribution.

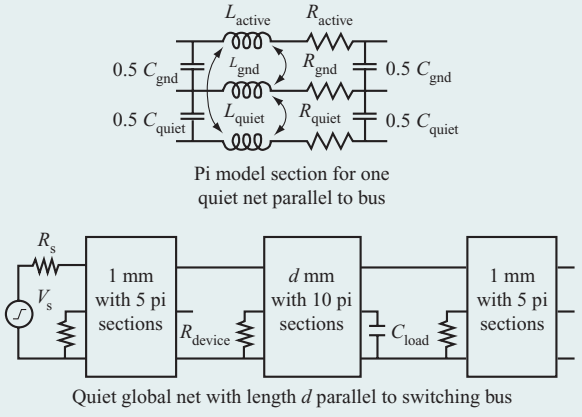

Figure 20

Example of circuit model used for calculating common-mode noise associated with simultaneous switching.

CPAM is an IBM power-analysis tool that uses IBM ACES[1] as its simulation engine, enabling it to analyze circuits of up to 500000 transistors. CPAM also takes into account logic orthogonality between inputs. With test and control inputs set at functional states, CPAM is first run to determine the power with only the clocks toggling and then rerun to determine the power with "random" input vectors applied to the circuit. During the second power analysis, CPAM controls the switching factor between consecutive input vectors to a user-specified value—typically 50%.

[1] Internally developed IBM design tools.

The worst-case switching factors were obtained by having the chip architects define worst-case simulation patterns. These patterns were run on both the core and full-chip logic models using the IBM TEXSIM logic simulator. Switching factors were obtained by simply counting the number of times each node toggled on a unit-by-unit basis and dividing by the number of simulation cycles.

The results of the power analysis were used to identify high-power regions on the chip for placement of decoupling capacitance, to keep the total chip power within the range suitable for packaging and cooling, and to support the package thermal design. For example, a chip power-density map constructed from the macro and unit power analysis was used by the packaging team to design the packaging and cooling system. Their prediction of worst-case packaged chip-junction temperatures based on a 140-W total chip power is shown in **Figure 19**. In this design, the hottest regions are in the data cache and fixed-point unit.

After the chip physical design was completed, a full chip analysis [34] was carried out to verify that the limits for average dc voltage drop and electromigration were not exceeded. These checks were performed by first rerunning CPAM on extracted netlists (with worst-case switching factors predicted by logic simulation) to generate average dc currents at every power and ground metal–silicon contact. Next, the full chip was divided into 250 overlapping regions, and the power grid was extracted for each region. Finally, the CPAM currents were applied to the extracted netlists, and a dc analysis was performed using an IBM fast linear circuit solver.

Because of the fast edge rates on switching signals and the wide data buses in the POWER4 design, common-mode noise was an important consideration in the power-distribution design. Common-mode noise circuit models, like the one depicted in **Figure 20**, were constructed using FASTHENRY [35] to generate the effective *RL* matrix of a quiet line in the midst of a wide, simultaneously switching bus. Circuit analyses with these models showed that a change was needed in the typical IBM microprocessor power-distribution design. Heretofore, the top-metal power distribution had always consisted of wide power/ground buses repeated along the C4 rows. Power distribution at the lower-level-metal levels, while not pinned to the C4 pitch, still used relatively wide power/ground buses repeated at wide intervals. While this design point had always satisfied dc requirements in earlier designs, our early circuit analyses quickly showed that it would not satisfy our common-mode noise requirements.

The solution was conceptually simple: Use more power buses to reduce the return-path inductance and resistance. This has the additional benefit of creating more quiet

wiring tracks, but required additional (scarce) wiring tracks. Some additional tracks were obtained by redesigning and qualifying a new rectangular metal C4 pad design, and, after considerable analysis and negotiation with unit and chip integrators, the power grid that came closest to meeting the dc-noise, common-mode-noise, quiet-wiring-track, and wirability requirements was a fourteen-wiring-track image. This is depicted in **Figure 21** along with predictions for worst-case common-mode noise. It devotes four wiring tracks to power and ground and ten wiring tracks to signal wiring. Because portions of the design were unable to strictly conform to this image, exceptions were allowed provided that some rough guidelines were followed: Power and ground should use at least 25% of the wiring tracks, and no more than ten signal wires were permitted between power buses.

Integration

The integration cycle begins with floorplanning at the respective level of hierarchy: unit, core, or chip. The netlist is built from the VHDL description through a compilation process utilizing both high-level (Hiasynth) and low-level (BooleDozer) synthesis. The resulting netlist is then imported into the Cadence design framework in the form of an autoLayout view. The design framework serves as the repository of the design data for all design elements: custom, RLM, unit, core, chip. The new autoLayout view is initialized with any previous floorplanning information.

Floorplanning

Once the floorplan is created, the integrator sets the block sizes and aspect ratios through a set of iterative moves. The floorplanning environment allows for rectilinear shapes on blocks to maximize area utilization. Size and aspect ratios are communicated to each block owner for feedback. Once block sizes and aspect ratios are established, pin assignment and track allocations are made.

Significant over-block routing is utilized to minimize wire length, reduce congestion, and improve timing. To facilitate over-block routing, detailed blockage contracts are established between adjoining upper (parent) and lower (child) members of the hierarchy. These contracts specify the exact routing tracks on each routing layer which are used by the parent and child. The child blockage is represented to the parent in the form of an abstract. Every floorplannable object has an abstract: custom, rlm, unit, core. The child uses an inverse image of the abstract (a cover) to perform internal routing. The cover is automatically derived from the abstract. If a child shell needs to share chip-level resources, an additional cover (the parent_cover) is needed which is a union of all of the chip infrastructure that appears above all instances

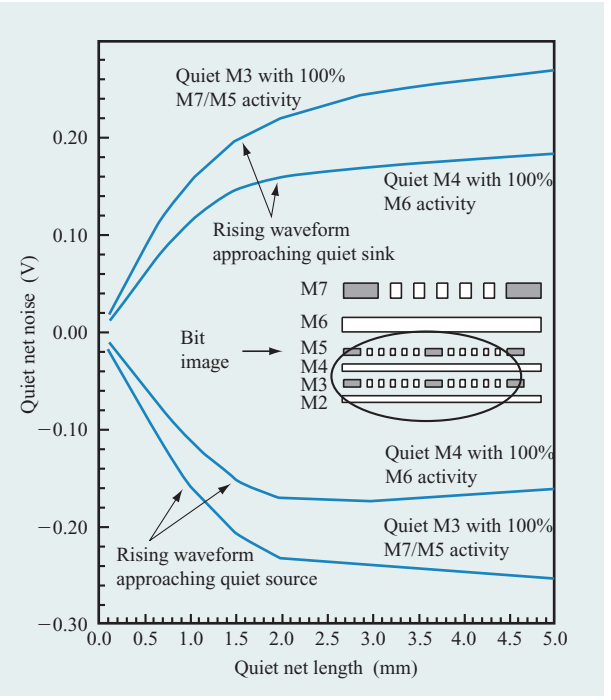

Figure 21

Example of common-mode noise simulation results. Noise on quiet wires is calculated for several different switching assumptions on nearby wires. Inset shows a cross section of the 14-bit image with power wires shaded.

of the child. This view is automatically generated at each level of hierarchy and propagated downward to the leaf cells. Additional constraints are also enforced in the abstract/cover to ensure smooth child/child and parent/child interaction. For example, the parent "owns" the routing track immediately outside the outline of the child for purposes of adjacent via rules. Therefore, the child's cover contains additional via blockages to prevent via placement on the track immediately inside the outline of the child. Once a blockage contract is established, pin assignment can be performed on the child cells. Pins are assigned only on routing tracks owned by the child. In this way, accessibility from within the child is guaranteed, and through-tracks assigned to the parent are not interrupted by child pins. Pins are predominately assigned at the perimeter of blocks. However, pin placement anywhere within the child outline is allowed. Pins must be accessible to the parent from the pin layer or any layer above the pin. Pins must be accessible to the child from the pin layer or any layer below. Via obstructions are placed above or below the pins in the respective cover or abstract to ensure that the parent and child do not conflict at a pin.

Architecturally significant buses received special attention during the floorplanning of units, core, and chip. Net length guidelines were established during floorplanning to control time-of-flight and slew rates of nets. The large size of most units meant that buses could not span the unit without intermediate buffers or latches. Therefore, the locations of buses, buffers, and latches were planned early in the process so that changes could be incorporated into the architecture. In addition, in areas where buses spanned large distances over a unit, buffer locations were established, and "holes" were opened in the unit to accommodate the buffer. The buffer or latch blockage and power pins are reflected in the parent_cover of the child. Nets which were not timing-sensitive or architecturally important were routed around large units, where they could be buffered and latched at appropriate intervals.

RLM build

The random logic macro (RLM) build methodology was designed as a complementary process to create high-performance synthesized control blocks for each level of the floorplan. The process that was defined uses iterative timing-driven design at all levels: synthesis, placement, clock insertion and optimization, routing, and post-placement/routing optimization. The flow uses a combination of internal synthesis tools and transformations, qplace for timing-driven placement, and wroute for timing-driven routing.

The RLM build process begins with synthesis of a netlist from VHDL using wire-load models derived from actual physical parameters of numerous completed RLMs. In this initial netlist, the clock network is not fully implemented. A local clock buffer (LCB) for each unique clock-phase combination is connected to all of the corresponding latches. In addition, a single clock control block (LCB driver) is connected to all LCBs. Next, an initial timing-driven placement is performed while ignoring the clock, scan, and miscellaneous control nets and with the LCB and LCB driver blocks placed at the origin. The placement gives the optimal data-path placement in order to achieve timing without consideration of the clock network. Clock network insertion is performed on the basis of this placement. During this phase, the correct number of LCBs are inserted and connected to the correct number of latches to meet clock-distribution constraints. An initial LCB placement is made, and all of the associated latches are placed in a placement region around the LCB. A second timing-driven placement is then performed, with the LCB locations fixed and the latches allowed to float within their respective regions. Next, routing is performed in two steps. First, the clock network from RLM input pins to the input pins of the LCBs receives a network of virtual pins and subnets to replace the original net. This allows for straight point–point routing from the input pins to the LCB. During this phase, the clock nets from the LCBs to the latches are also weighted to provide priority routing. Timing-driven routing is performed on the entire RLM, with the clock nets receiving priority over signal nets. At the conclusion of the route, the design can be modified with post-physical design synthesis transforms to optimize performance. The design can then be incrementally placed and routed to correct any changes.

Infrastructure and wiring

The chip infrastructure comprises all of the necessary design elements to distribute clock, power, and I/O to objects in the floorplan. These three portions of the design use the topmost levels of thick, wide metal to control *RC* delay, *IR* drop, and impedance. Since these three portions consume a large amount of wiring resource and in many cases compete for the same routing areas, they must be designed and optimized concurrently.

The infrastructure design process begins with signal I/O assignment based on the floorplan positions of critical components. Feedback is also incorporated from the package to modify the location of signal I/O. Next, an initial clock distribution is created with ideal buffer locations. The I/O trunk wires and an initial power distribution are created using the top thick metal levels LM and MQ. A process of iteration is used to create the final allocations for the signal I/O wires, clock distribution, and power. Once this iteration process is complete, the final determination of the location, widths, spaces, and pitches of the clock distribution, signal I/O trunks, and power distribution is complete. The remaining tracks on LM/MQ are now available for the engineered buses.

Engineered buses are buses which are defined to be architecturally critical in maintaining performance. They typically control topological problems caused by the need to span large floorplanned blocks. To improve the performance of these buses, they are routed on the thicker, lower-resistance LM/MQ wiring levels. They may, in addition, be specified to be wider than the nominal LM/MQ wire width. Wire codes are used to model these wires accurately in the chip timing environment. The list of engineered wires and the associated widths/spaces are derived from the wire codes to ensure that the engineered routes match the topologies estimated in timing.

To manage data volume and performance, a blockage map and pin model are abstracted from the autoLayout view of the design. At this point in the process, for nets that require extremely exact placement to control delay, slew rate, or coupled noise, trunk routes are placed with SKILL code. The trunk routes can be point-to-point, contain one turn, or contain one "T" junction. IC Craftsman** is now used to complete the routes from the trunks to associated pins, and to route all other engineered routes. IC Craftsman allows fully off-grid

routing of the engineered routes with extremely complex rules governing width, space, and neighboring nets. This creates the most efficient use of routing resources for both engineered and non-engineered routes. The engineered routes in POWER4 are shown in **Figure 22**; there are approximately 35 000 such routes out of a total of 120 000 nets at the chip level.

To complete the chip routing, a layout view of the engineered routes is abstracted and added to the chip autoPlaced view. Normal "non-engineered" routes are required to route around the blockages defined by the engineered routes. Non-engineered routes are routed on-grid. They can take the form of wide or normal-width wires. Because of the on-grid routing, wide wires are inherently less efficient in the use of routing resources. Wide wires are routed first, by default, by wroute. Other default-width wires are also priority-routed in groups based on criticality with the use of "selectNet" lists. Finally, all remaining noncritical nets are routed. The entire process of routing normal routes takes approximately 1.5 hours. This turnaround time was critical to ensure that rapid timing learning could be achieved with extracted routing information.

Global buffering of non-engineered nets is essential to control propagation delay, slew-rate degradation, and coupled noise on long nets. Buffering rules were established to screen nets that required buffering without the need to do a timing run. For example, a single-cycle net exceeding 3000 tracks in length required a buffer. Similar rules exist for multi-cycle and test nets. To facilitate rapid buffer insertion and timing, a set of tools were developed to work with buffers that were pre-placed at each level of the hierarchy. For example, the chip has approximately 450 32-bit buffer packs inserted as uniformly as possible throughout the floorplan. Initially, these buffer packs are inserted with all inputs grounded and the outputs floating. Two modes were used to accomplish most of the needed buffering. In the first, the exact topology of the buffered net was described in a file. The files would then be interpreted and the net buffered accordingly. In the second, a net could be buffered automatically by finding the shortest path from the source to all sinks by traversing available buffer packs. Since the global buffering solution is not defined in the VHDL, it is necessary to reapply the solution each time a chip is initialized with a new VHDL release. To facilitate this, a companion tool wrote the existing buffering topology to a file, such that it could be reapplied on subsequent releases.

Because of the number of nets at the chip level, rapid iteration to improve timing was necessary. The methodology was established to complete a timing iteration daily. This included a review of the previous timing run, buffer changes, a full chip global and final route, 3D extraction of the routed data, and a full chip-timing run.

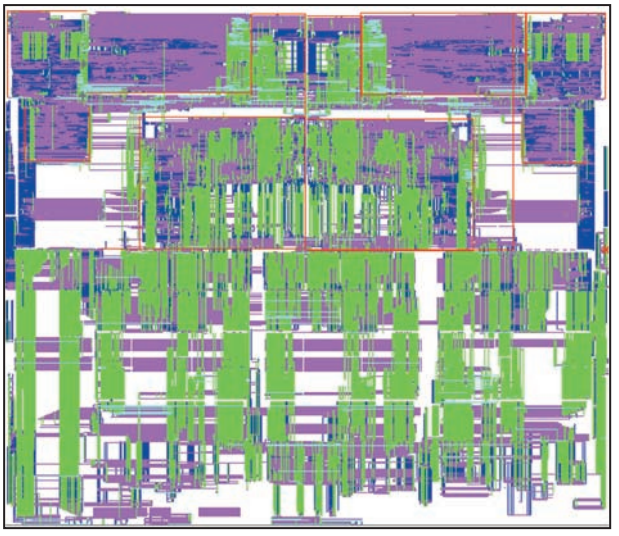

Figure 22

Engineered buses in POWER4. The green (vertical) routes are in the MQ (sixth) wiring level; the lavender (horizontal) ones are in the LM (seventh).

Checking strategy

The checking strategy was instrumental in ensuring that the final chip could be assembled with a minimum of problems. Because of the size of the chip, a large number of problems at the end of the design cycle would be too difficult to detect and fix. Therefore, the checking methodology was developed to treat each unit and the core as a "chiplet." In addition, a robust set of "methodology" checks were developed to ensure that all macros, units, and the core could be correctly integrated at the next level of hierarchy.

In order to check an entity as a chiplet, it is necessary to understand the environment in which the chiplet resides in the chip. To model this environment, the cover (routing contract with the parent) and the parent_cover (fixed chip-level infrastructure) were added to the unit for DRC and LVS verification. Since the covers contain blockage layers, as opposed to manufacturable shapes, a separate set of checks were included for spacing of manufacturable shapes to blockages of the same layer. No minimum-area checks were done on the blockage shapes, since they are not required to comply with area rules. Similarly, the blockage shapes were considered during the LVS run for purposes of determining shorts, since any manufacturable shape touching a blockage in a cover would be a short to a parent object.

A separate-methodology DRC deck was created to check additional design constraints above and beyond the design rules necessary for manufacturing specified in the DRC deck. These checks concentrated on ensuring the quality of the blocks as well as their ability to be integrated at the next level of hierarchy. The types of design constraints checked included ensuring that all manufacturable shapes are "one-half ground rule" (i.e., conform to a symmetric half of the shape definition, or "ground rule") for the boundary of the floorplan block, power buses are on the correct periodicity, and clock pins are in the correct track. Additional checks maintained the quality of the design for routing. For example, pins were checked to ensure that they were on grid and accessible from the same layer or the layer above.

Chip timing and extraction

Timing closure at frequencies above 1 GHz in large, complex system-on-a-chip designs requires both the capability for rapid iterative refinement and a high degree of concurrency among timing activities occurring at all levels of the design, including macro, unit, processor core, and chip. The scale and complexity of these timing tasks posed a major challenge to the POWER4 chip timing team. In addition, the transparent latches described earlier added to the difficulty of the timing task, with critical timing paths at the chip level frequently involving as many as 10 to 20 stages of transparent latches traversing multiple units. Finally, the timing methodology had to deal simultaneously with multiple frequency domains, in which certain parts of the chip operated at different multiples of the global clock frequency. Since the exact multiple of the clock frequency used in these regions was programmable, the timing analysis had to provide the worst-case timing for any selection of the multiple clock frequencies, for paths potentially traversing more than one frequency domain.

Hierarchical timing strategy

Timing closure was pursued concurrently by the circuit and logic designers at the macro level, by the unit team at the unit level, and by the global timing and integration team at the levels of the processor core, the memory subsystem (GPS), and the chip (cf. Figure 2). The timing methodology exploited the physical design hierarchy to facilitate a "divide and conquer" approach to timing closure. Because of the lack of hard timing boundaries at all levels of the hierarchy, this required frequent communication of updated timing contracts to reflect how design changes in one block affected the timing requirements of the other blocks in the design. This in turn drove the requirement for frequent timing iterations at the chip level. Timing contracts were determined hierarchically starting at the global chip level and then propagated down the hierarchy through the processor core and memory subsystem to the units and then on to the individual macros. These contracts were generated nightly from each chip-level timing run and consisted of input pin arrival times and slews, and output pin required times and lumped capacitive loads. Input arrival and output required times were both computed to distribute positive and negative slack so as to provide individual macros and units with contracts that, if achieved, would close timing globally. Capacitive loads were computed as C_{eff} (effective capacitance seen at the driver output) as opposed to C_{tot} (total lumped capacitance on the net) to model the effect of resistive shielding on long unit and global chip nets, and to optimize timing closure for minimum delay on wire-dominated paths. On the basis of these contracts, macro, unit, processor core, GPS, and chip timing all proceeded in parallel, passing information back and forth regularly to validate the ongoing work at all levels as timing was iterated toward global closure.

An important requirement of the hierarchical timing methodology was the ability to incorporate a mix of (sometimes inconsistent) data with differing accuracy/quality characteristics from different levels of the design hierarchy and from different regions of the design. This feature was essential, since different parts of the design proceeded toward completion at different rates. Early on in the design, chip-level timing was based on an initial chip floorplan, using time-of-flight delays for unit-to-unit routes estimated in ChipBench*, the floorplanning tool which was also the interface to the IBM static timing tool, EinsTimer. Closure of timing at this level was used to refine the floorplan and generate the first unit-timing contracts. Unit-level timing was based on estimated delays from the initial unit floorplans, with macro-level delays from hand-coded Delay Calculator Language (DCL) rules, or with lists of asserted times required for macro inputs, and times at which macro outputs were available. As the design proceeded, these simple macro timing rules were replaced by synthesized networks of standard-cell library books (for RLMs), and timing rules (DCMs) for custom macros, based either upon macro schematic descriptions or upon physical data extracted from layout. Estimated wires at different levels of the hierarchy were gradually replaced with extracted data from real routed wires, until finally the whole chip was timed with data extracted from the physical design. At intermediate stages in the design, it was therefore necessary to use a mix of estimated and/or incomplete data along with all the latest extracted data as shown in **Figure 23**. A normal mix of design data could include, for example, extracted SPICE[2] *RC* networks from actual global routes at the chip level, in combination

[2] SPICE: Software Process Improvement and Capability Emulation, an ISO standard simulation process.

with estimated unit-level and RLM wiring data. ChipBench included features to detect the nets or net segments with no extracted data and automatically provide an estimated extraction of the missing routes in SPICE *RC* format. The estimated unit and extracted global SPICE networks were then stitched together across the hierarchy into one complete network. This capability enabled the units to evaluate and tune their wiring solutions to inter-unit timing problems and address noise, slew, and buffering issues in the context of the global chip data. As wired units became available, their extracted SPICE *RC* networks were incrementally added to the global chip timing runs.

Wire-delay and coupled-noise analysis

For a large chip such as the POWER4, running at frequencies exceeding 1.3 GHz, it was clear that wire delay would be a critical factor in many cycle-limiting timing paths. This drove the requirement for accurate modeling of wire delay during all phases of the design process, from high-level design through to the end of physical design. Early in the design cycle, unit or global wiring parasitics were estimated on the basis of Steiner routing approximations and a "2½ D" extraction process of these estimated routes. At this stage, it became apparent that a large number of wires would have to be broken into segments and buffered to avoid excessive signal slews and *RC* delays. Many large banks of buffers were placed at various locations in the units, the core, and across the chip to facilitate rapid fixes for nets with problem slew rates. In addition, as the design progressed, a large number of buffers were placed individually for those signals with more critical timing requirements. Finally, for many critical nets, customized solutions were required to simultaneously solve particular delay, noise, and slew-rate challenges. Such custom engineering techniques could include the use of wider wire widths, extra spacing between wires, and special track assignments or semicustom wire pre-routing to minimize delay degradation due to capacitive coupling and simultaneous switching. To capture these effects early in the design, before physical implementation, special wire codes were assigned on a net-by-net basis and passed to ChipBench to allow modification of the estimated extraction process. More than 50 000 nets in the design had wire codes assigned, including 35 000 nets in pre-routed engineered buses. A total of 59 different wire codes were used, specifying all allowed combinations of width, spacing, and hostility.

As the design proceeded, physical data became available for various nets in the design, so that real extracted data could be used where available to replace the estimated data. At the custom macro level, all wiring parasitics were extracted from the macro layout, using a cover cell to simulate the (worst-case) effect of all possible global routes over the macro. This extraction data was used by the transistor-level timer and incorporated into the timing model for the macro. Nets inside RLMs and unit or chip nets were extracted using 3DX, an IBM proprietary tool developed for three-dimensional extraction [36]. A rapid turnaround time for extraction was achieved by exploiting the natural chip hierarchy. Unit wires were extracted down to the custom macro I/O pins, but included the nets inside the RLMs by flattening the physical hierarchy down to the RLM standard-cell library books. Again, a cover cell was used to provide an estimate of the effect of the wires in the upper levels of the hierarchy. The global wiring in the processor core was extracted as a separate hierarchical entity, as was the top-level chip global wiring. All of these extractions were independent and could proceed in parallel. The maximum run time for any one 3DX extraction was four hours, that being for the top-level chip global wires. By comparison, a flat extraction of the entire chip required 26 hours. In addition, with the hierarchical extraction capability, only the portions of the design that

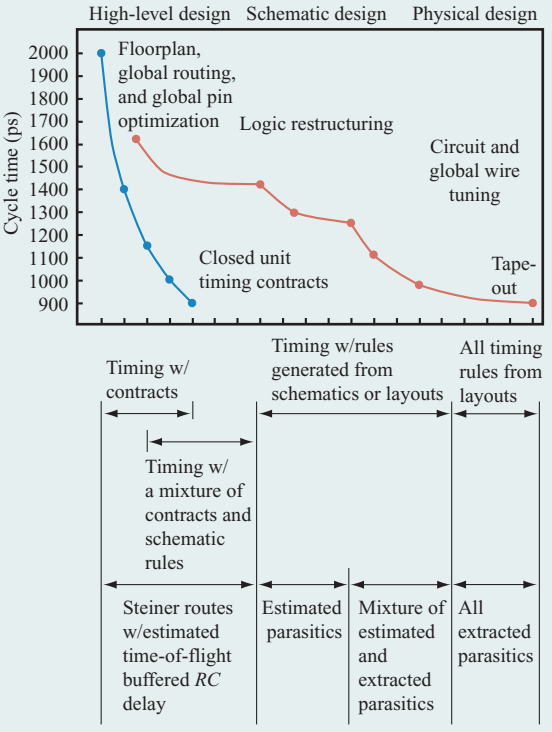

Figure 23

POWER4 timing closure progression illustrating the mixture of design data from high-level design and initial unit timing contract generation to final physical design and tape-out.

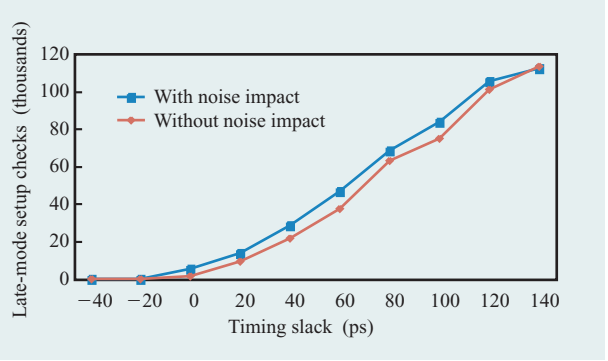

Figure 24

Illustration of the effect of the noise uplift on timing. POWER4 path delays are shown before and after taking into account the additional wire delay due to noise effects.

had actually changed from the previous run had to be re-extracted.

Every net in the design not contained inside a custom macro was modeled for EinsTimer as a distributed *RC* network in SPICE format, created by stitching together the segments across the hierarchical boundaries. Each net was analyzed from source to sinks and transformed by an AWE/RICE (asymptotic waveform evaluation/rapid interconnect circuit evaluation) Order 4 analysis into a near-end load used to determine the driving-block delay (C_{eff}), a PI model used to calculate near-end slew, and for each sink, a poles-and-residue representation of the transfer function used to determine the far-end slew and the *RC* delay of the wire to that sink. This analysis achieved agreement to within 5% of the delay predicted by SPICE circuit simulation of the extracted data.

Delay degradation due to capacitive coupling and simultaneous switching, referred to as "noise impact on timing," is a very significant effect for high-performance designs for technologies at and beyond the 0.18-μm generation. In addition to explicitly modeling noise impacts on timing using wire codes on estimated wires, it was necessary to have a methodology that analyzed the actual extracted routes in the context of the chip static-timing analysis. As soon as global routing data was available at the chip and processor core levels, explicit capacitive uplifts were calculated on a net-by-net basis. The input to this analysis included the extracted *RC* networks modeling the wires, timing windows specifying coincident edges from EinsTimer runs using non-uplifted wire extraction data, and information from the floorplan and the physical database specifying adjacency of net segments. Timing-window files were created from each daily chip timing run, and consisted of rising and falling edge times and slew rates for both early- and late-arriving signals on each net. From the physical data, pairs of physically adjacent victim and aggressor nets were chosen for analysis. Falling (rising)-edge data of an aggressor net was compared to rising (falling)-edge data at the sink of the victim net, and the timing overlap was determined by the slew rates of the nets with an assumed 100-ps "window of vulnerability." The percentage of timing overlap was then used to scale the side-to-side capacitance between the two nets in the extraction. That is,

$$\text{Cuplift} = \text{Ctop} + \text{Cbottom} + \text{Cleft}^*(1 + \text{left\_overlap}\%) + \text{Cright}^*(1 + \text{right\_overlap}\%).$$

The data extracted for the chip was postprocessed, and the capacitance on nets was increased on the basis of the above formula. This updated extracted data was then fed back into a chip timing run to determine which nets were experiencing unacceptable delay degradation due to noise coupling. This information was used to drive floorplanning and wiring changes in the next chip timing iteration, to ensure that the chip achieved its cycle time under actual operating conditions. **Figure 24** shows a comparison of final POWER4 path-delay histogram results using both extracted routes and extracted routes after postprocessing to include noise effect on timing uplifts.

Timing data and statistics

Timing closure on the design required rapid timing iteration with an overnight turnaround time in the presence of a very large amount of design data [2]. The complete timing/integration process is shown in **Figure 25**. The POWER4 design had a total of 4341 instances of macros of all types, instantiated from 1015 unique design blocks. (Table 2 shows the breakdown of instances and unique blocks by type.) In addition, there were approximately 100 000 buffers and inverters added to maintain slew rates at or below one third of the cycle time on long nets. In the case of custom and array macros, each unique macro had an associated timing rule (DCM) that described all paths through the macro from primary input to primary output, with input slew and output load sensitivities, as well as data and clock delay segments to each latch point internal to the macro. For dynamic circuits, each node where clock meets data is captured in the DCM, allowing visibility of all timing checks in the global timing analysis. DCMs for complex array macros were generated from a transistor-level timing analysis, modeling the SRAM cells with a gray-box description. RLMs were described at the gate level using the synthesized standard-cell netlist and the timing delay rules for each library book. Thus, in each chip timing run, every latch and every dynamic node was represented in the timing model, along with a total of 1.1 million nets in the

netlist. Of these, 121 713 were top-level chip global nets, and 21 711 were processor-core-level global nets. Against this model 3.5 million setup checks were performed in late mode at points where clock signals met data signals in latches or dynamic circuits. The total number of timing checks of all types performed in each chip run was 9.8 million. Depending on the configuration of the timing run and the mix of actual versus estimated design data, the amount of real memory required was in the range of 12 GB to 14 GB, with run times of about 5 to 6 hours to the start of timing-report generation on an RS/6000* Model S80 configured with 64 GB of real memory. Approximately half of this time was taken up by reading in the netlist, timing rules, and extracted *RC* networks, as well as building and initializing the internal data structures for the timing model. The actual static timing analysis typically took 2.5–3 hours. Generation of the entire complement of reports and analysis required an additional 5 to 6 hours to complete. A total of 1.9 GB of timing reports and analysis were generated from each chip timing run. This data was broken down, analyzed, and organized by processor core and GPS, individual unit, and, in the case of timing contracts, by unit and macro. This was one component of the 24-hour-turnaround time achieved for the chip-integration design cycle. **Figure 26** shows the results of iterating this process: A histogram of the final nominal path delays obtained from static timing for the POWER4 processor.

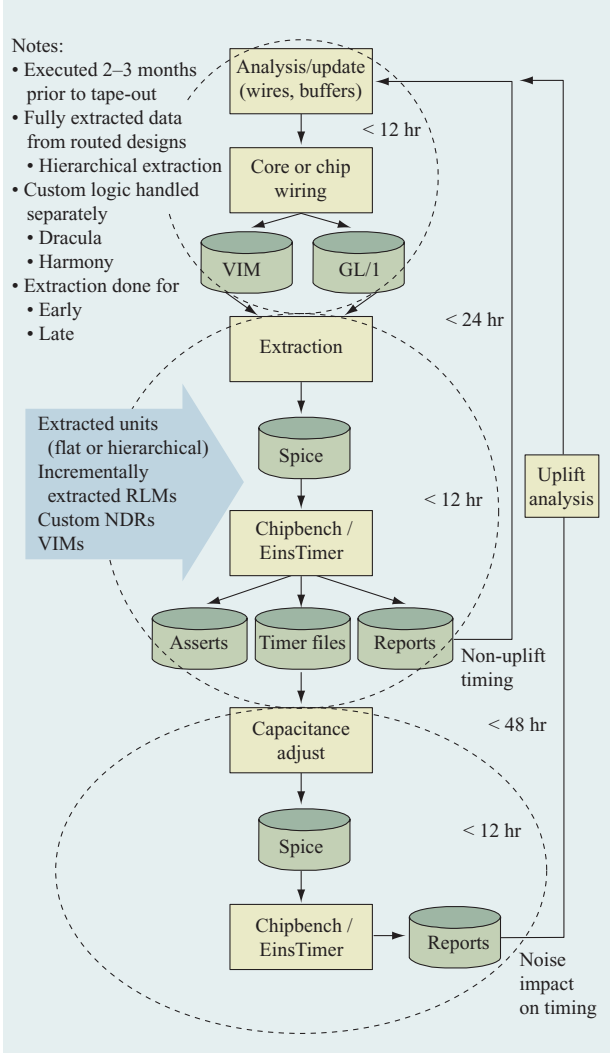

Figure 25

POWER4 timing flow. This process was iterated daily during the physical design phase to close timing.

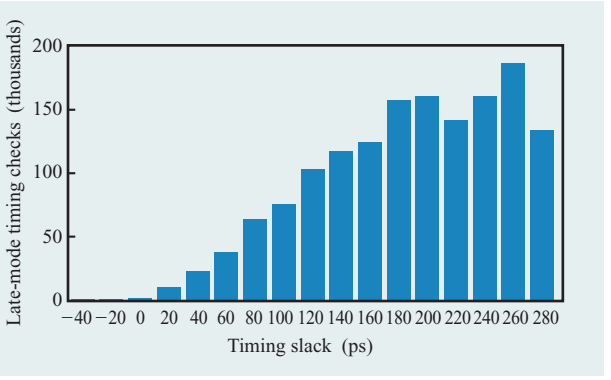

Figure 26

Histogram of the POWER4 processor path delays.

The POWER4 design includes LBIST and ABIST (Logic/Array Built-In Self-Test) capability to enable full-frequency ac testing of the logic and arrays. Such testing on pre-final POWER4 chips revealed that several circuit macros ran slower than predicted from static timing. The speed of the critical paths in these macros was increased in the final design. Typical fast ac LBIST laboratory test results measured on POWER4 after these paths were improved are shown in **Figure 27**.

Summary

The 174-million-transistor >1.3-GHz POWER4 chip, containing two microprocessor cores and an on-chip memory subsystem, is a large, complex, high-frequency chip designed by a multi-site design team. The performance and schedule goals set at the beginning of the project were met successfully. This paper describes the circuit and physical design of POWER4, emphasizing aspects that were important to the project's success in the areas of design methodology, clock distribution, circuits, power, integration, and timing.

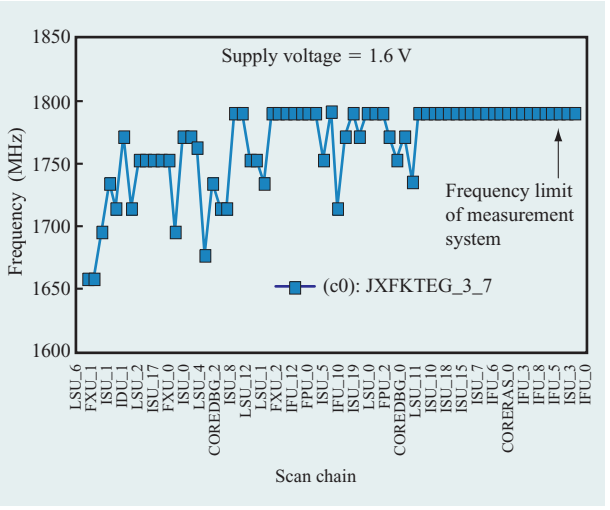

Figure 27

Typical fast results of LBIST (Logic Built-In Self Test) on-chip ac testing for POWER4. The maximum operational frequency is given for each scan chain. A scan chain comprises a group of circuit macros from a given region of the design.

Managing the complexity of the design represented a major challenge. Several approaches have been described that helped the POWER4 design to be completed correctly and to achieve its technical and schedule goals: The design was extensively partitioned (both horizontally and hierarchically) into blocks that could be designed concurrently. Capability was established for daily floorplan and timing updates to give feedback to the block designers. Updatable contracts were used between neighboring blocks for timing, block size/shape, pins, and wiring. Verification, checking, and audit tools were provided to ensure that the design was correct and that the design requirements were met by each block in the design. Spare circuits were provided to allow late changes to be made quickly and more easily. In addition, a common tools environment was developed that was used to keep the design synchronized among the multiple design sites.

A number of approaches have been described that have enabled the achievement of POWER4 performance goals. Results on them were presented: Top-down timing contracts were used that could be adjusted frequently on the basis of bottom-up design results; this enabled all aspects of the design to iteratively converge on the timing goals. Sophisticated clock network analysis and tuning minimized design margin required for clock uncertainty. Advanced circuit techniques were used to speed up critical paths, including the use of transparent latches, low-V_t devices, and circuit tuning tools. Key buses and critical long-wire paths were engineered early in the design and assigned fast wiring channels. Noise analysis was carried out, and the design was adjusted to minimize its impact.

Acknowledgments

The authors wish to acknowledge all of our colleagues on the POWER4 design team, as well as many others in IBM who have contributed to the POWER4 design. We would especially like to acknowledge the contributions of M. Amatangelo, C. Carter, S. Chu, J. DiLullo, P. Dudley, J. Eckhardt, J. Friedrick, I. Bendrihem, S. Neely, R. Kemink, P.-F. Lu, J. LeBlanc, G. Northrop, G. Nusbaum, D. Plass, G. Plumb, T. Rosser, S. Runyon, M. Scheuermann, P. Strenski, J. Venuto, J. Wagoner, R. Weiss, and S. Weitzel.

References

1. J. M. Tendler, J. S. Dodson, J. S. Fields, Jr., H. Le, and B. Sinharoy, "POWER4 System Microarchitecture," *IBM J. Res. & Dev.* **46,** 5–25 (2002, this issue).
2. C. J. Anderson, J. Petrovick, J. M. Keaty, J. Warnock, G. Nusbaum, J. M. Tendler, C. Carter, S. Chu, J. Clabes, J. DiLullo, P. Dudley, P. Harvey, B. Krauter, J. LeBlanc, P.-F. Lu, B. McCredie, G. Plum, P. Restle, S. Runyon, M. Scheuermann, S. Schmidt, J. Wagoner, R. Weiss, S. Weitzel, and B. Zoric, "Physical Design of a Fourth-Generation POWER GHz Microprocessor," *IEEE International Solid-State Circuits Conference*, *Digest of Technical Papers*, February 2001, pp. 232–233.
3. Keith Diefendorff, "POWER4 Focuses on Memory Bandwidth," *Microprocessor Report*, October 6, 1999, p. 11.
4. E. Leobandung, E. Barth, M. Sherony, S.-H. Lo, R. Schulz, W. Chu, M. Khare, D. Sadana, D. Schepis, R. Bolam, J. Sleight, F. White, F. Assaderaghi, D. Moy, G. Biery, R. Goldblatt, T.-C. Chen, B. Davari, and G. Shahidi, "High Performance 0.18 μm SOI CMOS Technology," *IEEE IEDM Tech. Digest*, pp. 679–682 (1999).
5. J. Petrovick and E. Seymour, "IBM POWER4 Chip Integration," presented at the Hot Chips 12 Conference, August 13–15, 2000; see *http://www.hotchips.org*.
6. K. L. Shepard, S. M. Carey, E. K. Cho, B. W. Curran, R. F. Hatch, D. E. Hoffman, S. A. McCabe, G. A. Northrop, and R. Seigler, "Design Methodology for the S/390 Parallel Enterprise Server G4 Microprocessors," *IBM J. Res. & Dev.* **41,** 515–547 (1997).
7. A. Bertolet, K. Carpenter, K. Carrig, A. Chu, A. Dean, F. Farraiolo, S. Kenyon, D. Phan, J. Petrovick, G. Rodgers, D. Willmott, T. Bairley, T. Decker, V. Girardi, Y. Lapid, M. Murphy, and R. Weiss, "A Pseudo-Hierarchical Methodology for High-Performance Microprocessor Design," *Proceedings of the International Symposium on Physical Design*, April 1997, pp. 124–129.
8. G. P. Rodgers, I. G. Bendrihem, T. J. Bucelot, B. D. Burchett, and J. C. Collins, "Infrastructure Requirements for a Large-Scale, Multi-Site VLSI Development Project," *IBM J. Res. & Dev.* **46,** 87–95 (2002, this issue).
9. P. Dudley and G. Rodgers, "Multi-Site Microprocessor Auditing and Data Management Methodology," presented

at the International Cadence UserGroup Conference, September 1999.
10. D. Brand, R. Damiano, L. van Ginneken, and A. Drumm, "In the Driver's Seat of BooleDozer," *Proceedings of the International Conference on Computer Design*, 1994, pp. 518–521.
11. K. L. Shepard, V. Narayanan, and R. Rose, "Harmony: Static Noise Analysis of Deep Submicron Digital Integrated Circuits," *IEEE Trans. Computer-Aided Design Int. Circuits & Syst.* **18,** 1132–1150 (1999).
12. A. Kuehlmann, A. Srinivasan, and D. P. LaPotin, "Verity—A Formal Verification Program for Custom CMOS Circuits," *IBM J. Res. & Dev.* **39,** 149–165 (January/March 1995).
13. S. Kundu, "GateMaker: A Transistor to Gate Level Model Extractor for Simulation, Automatic Test Pattern Generation and Verification," *Proceedings of the IEEE International Test Conference*, 1998, pp. 372–381.
14. P. J. Restle, T. G. McNamara, D. A. Webber, P. J. Camporese, K. F. Eng, K. A. Jenkins, D. N. Allen, M. J. Rohn, M. P. Quaranta, D. W. Boerstler, C. J. Alpert, C. A. Carter, R. N. Bailey, J. G. Petrovick, B. L. Krauter, and B. D. McCredie, "A Clock Distribution Method for Microprocessors," *IEEE J. Solid-State Circuits* **36,** 792–799 (May 2001).
15. P. J. Restle, "Technical Visualizations in VLSI Design," *Proceedings of the 38th ACM/IEEE Design Automation Conference*, Paper 31.1; see *http://dac.com*.
16. P. J. Camporese, A. Deutsch, T. G. McNamara, P. Restle, and D. Webber, "X-Y Grid Tree Tuning Method," U.S. Patent 6,205,571, March 20, 2001.
17. P. J. Restle, K. A. Jenkins, A. Deutsche, and P. W. Cook, "Measurement and Modeling of On-Chip Transmission-Line Effects in a 400 MHz Microprocessor," *IEEE J. Solid-State Circuits* **33,** 662–665 (April 1998).
18. A. Deutsch, G. Kopcsay, P. Restle, H. Smith, G. Katopis, W. Becker, P. Coteus, C. Surovic, B. Rubin, R. Dunne, T. Gallo, K. Jenkins, L. Terman, and R. Dennard, "When Are Transmission-Line Effects Important for On Chip Interconnections?," *IEEE Trans. Microwave Theor. Techniques* **45,** 1836–1846 (October 1997).
19. B. McCredie, J. Badar, R. Bailey, P. Chou, C. Carter, D. Dreps, J. Eckhardt, D. Ervin, M. Floyd, A. Haridass, D. Heidel, M. Immediato, J. Keaty, B. Krauter, J. LeBlanc, L. Leitner, D. Malone, D. Mikan, Jr., M. Nealon, J. Petrovick, D. Plass, K. Reick, P. Restle, R. Robertazzi, T. Skergan, K. Stawiasz, H. Stigdon, J. Vargus, and J. Warnock, "A 1 GHz POWER4 Testchip Design," presented at Hot Chips 11, Symposium on High-Performance Chips, Palo Alto, CA, August 1999.
20. L. Sigal, J. D. Warnock, B. W. Curran, Y. H. Chan, P. J. Camporese, M. D. Mayo, W. V. Huott, D. R. Knebel, C. T. Chuang, J. P. Eckhardt, and P. T. Wu, "Circuit Design Techniques for the High-Performance CMOS IBM S/390 Parallel Enterprise Server G4 Microprocessor," *IBM J. Res. & Dev.* **41,** 489–503 (1997).
21. K. Bernstein and N. Rohrer, *SOI Circuit Design Concepts*, Kluwer Academic Publishers, New York, 2000, Ch. 3.
22. P.-F. Lu, T. Ching, J. J. Chuang, L. F. Wagner, C. M. Hsieh, J. B. Kuang, L. L. C. Hsu, M. M. Pellela, Jr., S. F. S. Chu, and C. J. Anderson, "Floating-Body Effects in Partially Depleted SOI CMOS Circuits," *IEEE J. Solid-State Circuits* **32,** 1241–1253 (1997).
23. S. K. H. Fung, N. Zamdmer, I. Yang, M. Sherony, Shih Hsieh Lo, L. Wagner, T. C. Chen, G. Shahidi, and F. Assaderaghi, "Impact of the Gate-to-Body Tunneling Current on SOI History Effect," *Proceedings of the IEEE International SOI Conference*, October 2000, pp. 122–123.
24. A. G. Aipperspach, D. H. Allen, D. T. Cox, N. V. Phan, and S. N. Storino, "A 0.2-μm, 1.8-V, SOI, 550-MHz, 64-b PowerPC Microprocessor with Copper Interconnects," *IEEE J. Solid-State Circuits* **34,** 1430–1435 (November 1999).
25. D. Stasiak, J. Tran, F. Mounes-Toussi, and S. Storino, "A 2nd Generation 440 ps SOI 64b Adder," *IEEE IEDM Tech. Digest*, pp. 288–289, 465 (February 2000).
26. Ching-Te Chuang, Pong Fe Lu, and C. J. Anderson, "SOI for Digital CMOS VLSI: Design Considerations and Advances," *Proc. IEEE* **86,** 689–720 (April 1998).
27. K. L. Shepard and D.-J. Kim, "Static Noise Analysis for Digital Integrated Circuits in Partially-Depleted Silicon-On-Insulator Technology," *Proceedings of the ACM/IEEE Design Automation Conference*, June 2000, pp. 239–242.
28. C. Visweswariah and A. R. Conn, "Formulation of Static Circuit Optimization with Reduced Size, Degeneracy, and Redundancy by Timing Graph Manipulation, *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, November 1999, pp. 244–251.
29. A. R. Conn, I. M. Elfadel, 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," *Proceedings of the ACM/IEEE Design Automation Conference*, June 1999, pp. 452–459.
30. G. Northrop and P.-F. Lu, "A Semi-Custom Design Flow in High-Performance Microprocessor Design," *Proceedings of the 38th ACM/IEEE Design Automation Conference*, Paper 27.2, June 2001; see *http://dac.com*.
31. Huey Ling, "High-Speed Binary Adder," *IBM J. Res. & Dev.* **25,** 156–166 (1981).
32. C. T. Chuang and R. Puri, "SOI Digital CMOS VLSI—A Design Perspective," *Proceedings of the ACM/IEEE Design Automation Conference*, June 1999, pp. 709–714.
33. B. Curran, P. Camporese, S. Carey, Y. Chan, Y.-H. Chan, R. Clemen, R. Crea, D. H. Hoffman, T. Koprowski, M. Mayo, T. McPherson, G. Northrop, L. Sigal, H. Smith, F. Tanzi, and P. Williams, "A 1.1GHz First 64b Generation Z900 Microprocessor," *IEEE IEDM Tech. Digest*, pp. 238–239, 452 (February 2001).
34. J. Scott Neely, Howard H. Chen, Steven G. Walker, James Venuto, and Thomas Bucelot, "CPAM: A Common Power Analysis Methodology for High-Performance VLSI Design," *Proceedings of the 9th Topical Meeting on the Electrical Performance of Electronic Packaging*, 2000, pp. 303–306.
35. Mattan Kamon, Michael J. Tsuk, and Jacob K. White, "FASTHENRY: A Multipole-Accelerated 3-D Inductance Extraction Program," *IEEE Trans. Microwave Theor. Techniques* **42,** 1750–1758 (September 1994).
36. R. M. Averill III, K. G. Barkley, M. A. Bowen, P. J. Camporese, A. H. Dansky, R. F. Hatch, D. E. Hoffman, M. D. Mayo, S. A. McCabe, T. G. McNamara, T. J. McPherson, G. A. Northrop, L. Sigal, H. Smith, D. A. Webber, and P. M. Williams, "Chip Integration Methodology for the IBM S/390 G5 and G6 Custom Microprocessors," *IBM J. Res. & Dev.* **43,** 681–706 (1999).

Received June 21, 2001; accepted for publication January 8, 2002

James D. Warnock *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (jwarnock@us.ibm.com).* Dr. Warnock received the Ph.D. degree in physics from the Massachusetts Institute of Technology in 1985. Since then he has been at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, working on advanced bipolar and CMOS silicon technologies, and more recently in the area of circuit design for high-performance digital microprocessors. This included work on the S/390 G4 processor and also on the POWER4 chip, where he was the circuit design team leader. Currently Dr. Warnock is working on the Sony/Toshiba/IBM Broadband Engine, a leading-edge design in advanced IBM SOI technology. Dr. Warnock has experience in process technology and device design as well as SOI digital circuit design; he has authored or co-authored more than 160 conference or journal papers.

John M. Keaty *IBM Microelectronics Division, 11400 Burnet Road, Austin, Texas 78758 (keaty@us.ibm.com).* Mr. Keaty received a B.S. degree in mathematics from the State University of New York at Plattsburgh in 1974, and an M.A. degree in mathematics in 1977 and an M.S. degree in computer science in 1980, both from the University of Wisconsin at Madison. He then joined the IBM Microelectronics Division in Burlington, Vermont, to work on automated diagnostic systems for semiconductor logic products. Since that time he has worked in ASIC Product Development on several CMOS logic families and also on Industry Standard Microprocessor Development in the IBM Microelectronics Division before transferring to the IBM Server Group in Austin, Texas, in 1996. There Mr. Keaty was responsible for timing of the Spinnaker processor in the POWER4 system. He is currently a Senior Engineer in the IBM Microelectronics Division STI Design Center, responsible for global integration of next-generation broadband processors.

John Petrovick Mr. Petrovick was a Senior Technical Staff Member in the POWER4 custom microprocessor design group in Austin, Texas. He received a B.S. degree in electrical engineering from Arizona State University in 1983, and an M.S. degree in electrical and computer engineering from the University of Massachusetts in 1985. In 1985 he joined IBM Microelectronics, where he was responsible for circuit design and physical design methodologies for ASIC products. He held a variety of positions related to the physical design and design methodologies of custom X86 and RISC microprocessors before leaving IBM in 2001.

Joachim G. Clabes *IBM Enterprise Systems Group, 11400 Burnet Road, Austin, Texas 78758 (clabes@us.ibm.com).* Dr. Clabes received a Ph.D. degree in physics from the Technical University of Clausthal, Germany, in 1978. He was an assistant professor in the Institute of Solid State Physics at the University of Hannover, Germany, from 1978 to 1983, working on semiconductor surface and interface phenomena. From 1980 to 1981, he was a visiting scientist at the IBM Thomas J. Watson Research Center, working on metal/silicon interfaces. He joined the Research Center in 1983 in the Materials Analysis Department and collaborated on a wide range of materials science and advanced development projects within IBM. In 1993 Dr. Clabes joined the microprocessor development program, initially working as a circuit designer on S/390 G4 and subsequently as unit circuit leader on Pulsar and POWER4. Since moving to the Enterprise Systems Group in 2000, he has been the circuit design leader for POWER5. His research interests are in high-performance circuits, design methodology, and power-sensitive design concepts.

Charles J. Kircher *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (kircher@us.ibm.com).* Dr. Kircher is Manager of Advanced RISC Circuit Design at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York. He received a Ph.D. degree in electrical engineering from Northwestern University and joined IBM in 1967. He has worked on and managed a variety of projects in Si CMOS, Josephson tunnel junction, and GaAs technologies, investigating devices for high-speed logic, lasers, and related aspects of device physics and technology. Dr. Kircher is an author or coauthor of numerous publications; he holds several patents and IBM awards in these fields. Since 1997 he has managed the Research teams working on circuit and physical design for the POWER4 microprocessor.

Byron L. Krauter *IBM Enterprise Systems Group, 11400 Burnet Road, Austin, Texas 78758 (krauter@us.ibm.com).* Dr. Krauter received the B.S. degree in physics and mathematics and the M.S. degree in electrical engineering from the University of Nebraska, Lincoln, in 1976 and 1978, respectively. He received the Ph.D. degree in electrical engineering from the University of Texas, Austin, in 1995. He has been with IBM since 1979 and is currently working in a VLSI CAD tools development group in Austin, Texas. Dr. Krauter's research interests include sparse boundary element method (BEM) techniques for inductance and capacitance extraction and on-chip interconnect analysis.

Phillip J. Restle *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (restle@us.ibm.com).* Dr. Restle received a B.A. degree in physics from Oberlin College in 1979, and a Ph.D. degree in physics from the University of Illinois at Urbana in 1986. He then joined the IBM Thomas J. Watson Research Center, where he initially worked on CMOS parametric testing and modeling, CMOS oxide-trap noise, package testing, and DRAM variable retention time. Since 1993 he has concentrated on tools and designs for VLSI clock distribution networks contributing to ten IBM server microprocessors, as well as high-performance ASIC designs. Dr. Restle received IBM awards for the S/390 G4, G5, and G6 microprocessors in 1997, 1998, and 1999 and for VLSI clock distribution design and methodology in 2001. He holds four patents, has written 20 papers, and has given keynotes, invited talks, and tutorials on clock distribution, high-frequency on-chip interconnects, and technical visualizations in VLSI design.

Brian A. Zoric *IBM Enterprise Systems Group, 11400 Burnet Road, Austin, Texas 78758 (bzoric@us.ibm.com).* Mr. Zoric received his B.S. degree in electrical engineering from the University of Pittsburgh in 1990. He joined IBM that same year and has worked in the areas of CMOS VLSI design and design methodologies. Mr. Zoric is currently managing a team of engineers in support of the POWER4 design methodology.

Carl J. Anderson *IBM Enterprise Systems Group, 11400 Burnet Road, Austin, Texas 78757 (cja@us.ibm.com).* Dr. Anderson received his B.S. degree in physics from the University of Missouri in 1974 and his Ph.D. degree, also in physics, from the University of Wisconsin in 1979. He joined the IBM Research Division in 1979, initally working on circuit design, package design, and testing for the Josephson superconducting computer program. From 1983 to 1992 he worked in design and fabrication on the GaAs optoelectronics project. In 1992 Dr. Anderson became the circuit design manager of the G4 S/390 project, and was responsible for the first implementation of the S/390 high-end mainframe in custom CMOS technology (older designs had been in bipolar technology). In 1997 he moved to IBM Austin and began work on the POWER4 microprocessor. Dr. Anderson managed and was responsible for the physical design of the POWER4 microprocessor.