IBM POWER6 microprocessor physical design and design methodology

R. Berridge
R. M. Averill III
A. E. Barish
M. A. Bowen
P. J. Camporese
J. DiLullo
P. E. Dudley
J. Keinert
D. W. Lewis
R. D. Morel
T. Rosser
N. S. Schwartz
P. Shephard
H. H. Smith
D. Thomas
P. J. Restle
J. R. Ripley
S. L. Runyon
P. M. Williams

The IBM POWER6™ microprocessor is a 790 million-transistor chip that runs at a clock frequency of greater than 4 GHz. The complexity and size of the POWER6 microprocessor, together with its high operating frequency, present a number of significant challenges. This paper describes the physical design and design methodology of the POWER6 processor. Emphasis is placed on aspects of the design methodology, technology, clock distribution, integration, chip analysis, power and performance, random logic macro (RLM), and design data management processes that enabled the design to be completed and the project goals to be met.

1. Introduction

The IBM POWER6* microprocessor (**Figure 1**) powers the new IBM iSeries* and pSeries* systems. The microprocessor is fabricated in 65-nm silicon-on-insulator (SOI) technology [1] and operates at frequencies of more than 4 GHz. The microprocessor is a 13-FO4[1] design containing more than 790 million transistors, 1,953 signal I/Os, and more than 4.5 km of wire on ten copper metal layers. A robust multidomain power distribution network operates field effect transistors (FETs) at several operating voltages, with most of the chip running nominally at 1.15 V. Several frequencies of the POWER6 microprocessor sub-block (e.g., core and nest) were measured with multiple clock meshes using small skew clock grids. Several types of FETs, such as those with high and low threshold voltages, were utilized to balance power and performance. The complexity, size, high operating frequency, technology challenges, and power restrictions greatly exceeded those of earlier POWER* microprocessors.

The design methodology for the POWER6 microprocessor was completely enhanced in order to achieve a high-quality design. Innovations in design data management were key to obtaining a productive multisite design team. The POWER6 processor design methodology, together with a tight schedule, introduced challenges to the multisite design team that were beyond those of earlier POWER microprocessors. This paper describes these challenges in more depth as well as the results in the areas of design methodology, technology, clock distribution, integration, chip analysis, power and performance, random logic macro (RLM), and design data management processes.

[1]*Fanout of 4* (FO4) is a technology-independent metric used to describe the amount of logic that can be used between latches. For example, 1 FO4 is the amount of time it takes a signal to propagate through a gate driving four gates of equal size.

2. Design methodology overview

The POWER6 microprocessor design methodology is based largely on the design methodology of the IBM POWER4* processor [2] and the IBM eServer* z900 server [3]. For adequate execution time, the POWER6 microprocessor was designed hierarchically and each level in the hierarchy was designed concurrently. This is described in more detail in the section on the POWER6 physical design and chip integration methodology. Further, advancements in methodology allowed the design to remain in the pre-layout design phase much longer than in previous projects in order to optimize logic and floorplan and then to quickly execute layout implementation with relatively few surprises (see Section 9). In addition, several advances over previous projects were made by the multisite design team on auto-design cleanup and optimization both pre-layout and

0018-8646/07/$5.00 © 2007 IBM

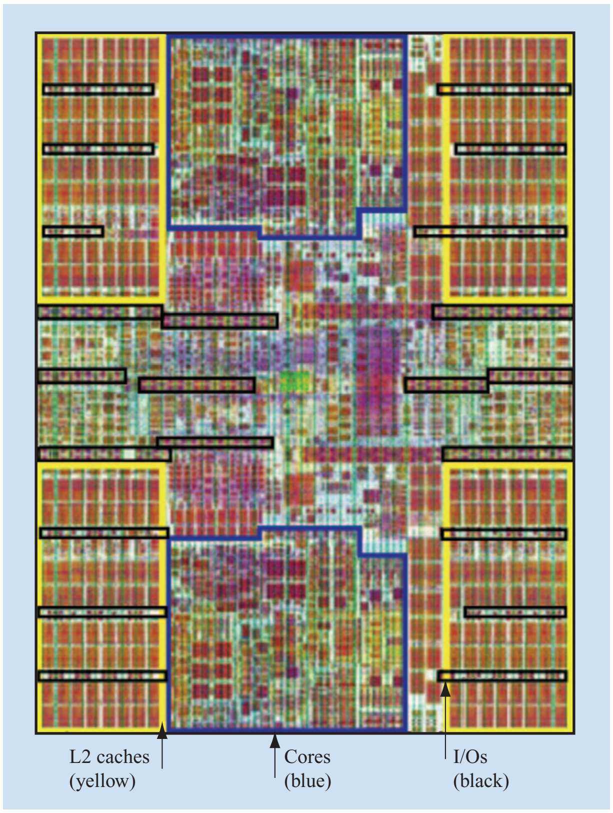

Figure 1

Floorplan of the POWER6 microprocessor.

post-layout. **Figures 2(a)** and **2(b)** depict the POWER6 processor methodology for pre-layout and post-layout design closure, respectively. **Table 1** describes the acronyms used in Figure 2.

The transistor count of the POWER6 processor design increased by about a factor of 3 and 4 over that of the POWER5* and POWER4 processor designs, respectively. This required enhanced tools and methodology so that the much larger design could be handled. Productivity enhancements were critical to achieving the design goals. These included enhancing the tools in order to increase design automation (DA), improve their accuracy, analyze designs automatically rather than manually, reduce the number of false errors, and manually review items that required analysis while reducing both runtime and the number of iterations required to complete the design. In addition, these productivity enhancements required the development of an IBM flow manager tool, the TaskManager, to manage the flows shown in Figure 2.

The logistical and communication challenges associated with an increasingly global design team required more robust enhancements in the integration space. Enhancements throughout the methodology were required in order to improve our ability to communicate and control changes at the interfaces between macros. This is described in more detail in Section 9.

Power and performance optimization is becoming more critical. A process was created to evaluate power and performance in the POWER6 microprocessor throughout the design (see Section 7). The design team incorporated multiple voltage levels and slower meshes in key sections of the design. This required the creation and verification of multiple power distribution and clock grids. Many of the tools had to be enhanced or created to accommodate this design style. Meeting the very aggressive cycle time of this design while minimizing the power requirements required significant improvements to the timing analysis process; this information was then exploited by other tools in order to reduce power in the non-timing-critical paths. (See the sections on circuit optimization and post-layout circuit optimization for performance and power.)

The smaller lithographic geometries required enhancements to the electrical analysis and physical design processes of earlier IBM microprocessors. For example, signal electromigration (EM) analysis was required to ensure hardware reliability. (Details on other analysis tools are presented in the section "Chip analysis closure" and its associated subsections.) On the physical design front, better estimation of wire parasitics was required to predict non-delay-scaling wires. This is described in more detail in the next section.

Custom pre-layout advancements

Since engineering changes can be prevalent during the early stages of the design and much time and effort are required to produce a custom layout, the pre-layout schematic design activities had to consider the physical implementation in order to minimize the number of changes that result when the actual physical layout and wiring data becomes available. Previous IBM microprocessor projects [2, 3] utilized technology device models, device parasitic models, and wire models when building schematics. This allowed the designers to perform multiple analyses at the schematic level before designing the physical layouts. Given the impacts of resistance · capacitance (RC) delays due to smaller wire geometries, wire model engineering at the schematic level was a source of inaccuracy. This was addressed by the development of pre-placed layout components with Steiner representations of interconnects between components. Key parameters placed on components were added within the schematic. Components were placed in a pre-layout using the IBM placement by instance parameter (PIP). This pre-layout was analyzed using the IBM Steiner estimated parasitics (STEP), which spliced

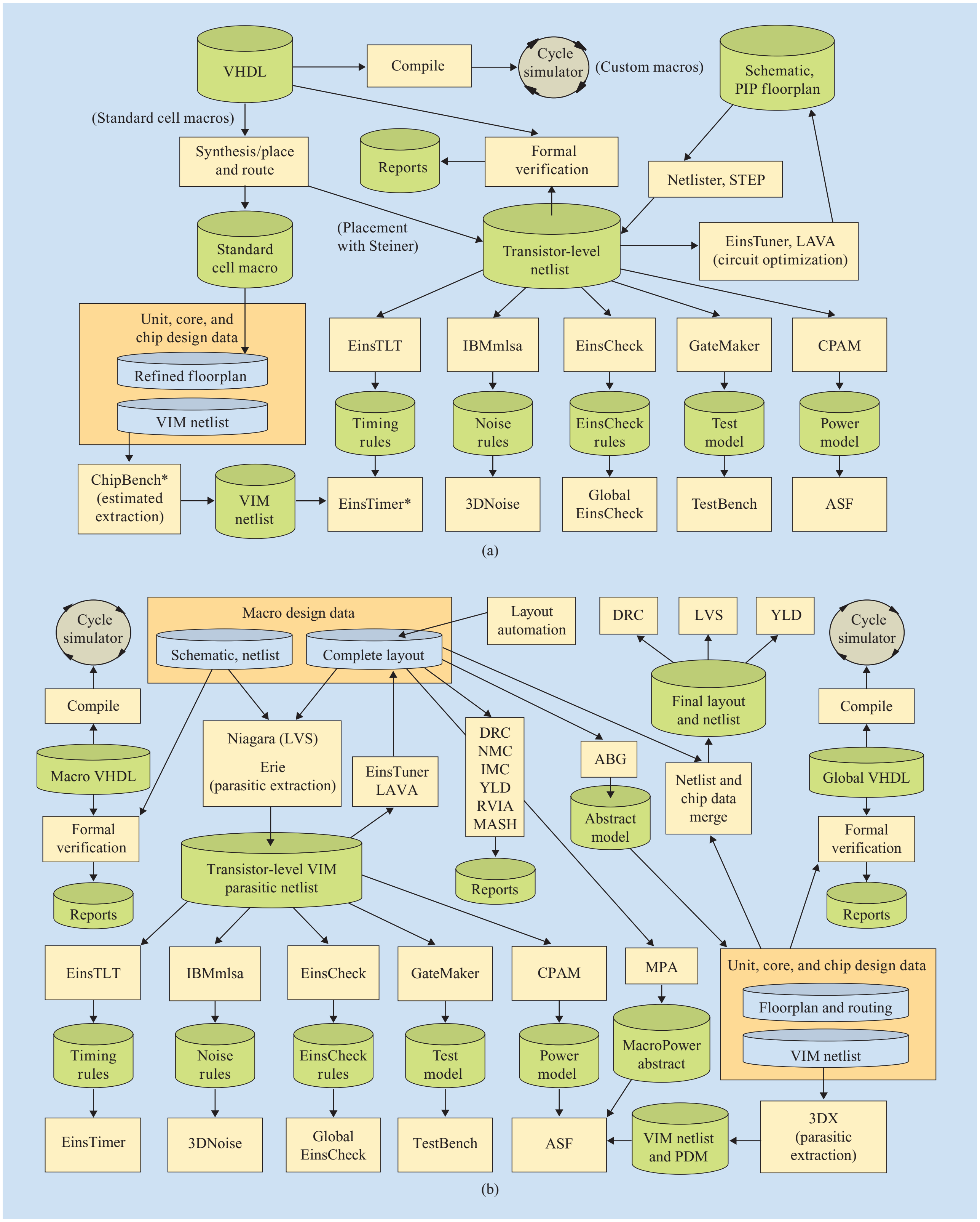

Figure 2

Macro- and global analysis-based (a) schematic and (b) physical design of the POWER6 microprocessor.

Table 1 Terms, abbreviations, and acronyms for Figure 2.

Tool name	*Full name*	*Description*
VHDL	(Very high-speed integrated circuit [VHSIC] Hardware Description Language)	Modeling language to describe logical functionality of a system
PIP	Placement by instance parameter	Custom macro cell placer
STEP	Steiner estimated parasitics	Steiner wire parasitic estimator
IBM EinsTuner	—	Circuit tuning tool
LAVA	Leakage avoidance and analysis	FET type switcher tool
VIM	Very-large-scale integration (VLSI) instance model	—
IBM EinsTLT	Transistor-level timer	Transistor-level timing tool
IBMmlsa	IBM macro-level signal analysis	Transistor-level functional noise tool
IBM EinsCheck	—	Transistor-level electrical checking tool
GateMaker	—	Transistor-level test model generation tool
CPAM	Common power analysis methodology	Power, voltage drop, voltage rail electromigration checker
IBM ChipBench	—	Chip viewing tool
IBM EinsTimer	—	Chip timing tool
3DNoise	—	Chip functional noise tool
Global EinsCheck	—	Chip electrical checking tool
TestBench	—	Chip testability tool
ASF	Austin linear simulator flow	Chip voltage drop and voltage rail electromigration checker
DRC	Design rule checking	Technology rule checking
LVS	Layout versus schematic	Layout and schematic equivalence tool
YLD	Yield	Yield checking tool
Niagara	—	Shape environment
Erie	Efficient rapid integrated extraction	Macro extraction tool
NMC	Niagara methodology checks	Hierarchy checker
IMC	Integration methodology checks	Hierarchy checker
RVIA	Redundant via generation	Auto via insertion
MASH	Migration assist shape handling	Layout shapes manipulator
ABG	Abstract generator	Hierarchy contract management tool
MPA	Macro power grid abstract	Macro power grid extractor
3DX	3D extraction	Chip wire extractor

technology-specific wire models into the schematic netlist.[2] Among the more accurately placed models in netlist, downstream analysis tools were more effective.

Circuit optimization

The IBM EinsTuner circuit tuning tool improved timing slack or performance by equally weighting many of the critical paths simultaneously while attempting to push the entire slack into positive territory with the IBM total positive slack (TPS) mode. Additionally, IBM free area recovery (FAR) within the EinsTuner tool included the optimization of noncritical paths in which circuits were reduced in size without migrating into critical or design-limiting territory. Essentially, this mode optimized these circuits for area recovery but, most importantly, reduced gate width by reducing the overall circuit leakage or dc power content of the macro design.

[2]A *netlist* describes connectivity in an electronic design.

The IBM LAVA (leakage avoidance and analysis) application was added to exploit additional device types [high-voltage threshold (VT) and low-VT] for both custom and standard cell designs. High-VT devices were substituted for regular-VT devices in very noncritical circuit paths in order to minimize leakage, whereas low-VT devices were incorporated into the design in highly critical circuit paths for increased performance gain while maintaining low leakage levels.

The combination of the EinsTuner and the LAVA applications, coupled with parameterized common components, allowed designers to optimize their pre-layout designs.

Post-layout circuit optimization for performance and power

As mentioned above, the RC delay became a larger fraction of overall delay in previous processors, so the EinsTuner had to be made operational on parasitic extracted netlist. The circuit sensitivity simulation engine utilized by EinsTuner was typically limited to capacitive-loaded networks because of runtime implications. Given this limitation, a method was developed to measure the RC delay between components within the IBM transistor-level timing (EinsTLT) tool [4] and abstract the results to EinsTuner for circuit optimization. With this method, the IBM linear wire model (LWM) enabled EinsTuner to take into account the RC delay during any circuit optimization mode, but more importantly, optimization now was made possible on both schematic and layout-based netlists.

With EinsTuner capable of optimizing layout-based netlists, opportunities for optimization were made possible even late in the design phase. If formulated, specific layout constraints could be passed to EinsTuner to adjust or optimize circuit width for additional performance. During any parallel design development strategy and late in the design cycle, significant optimization changes could result in the need to modify electrical or physical areas of the designs. To assist the designer with additional performance gain and minimal layout destruction, cells that were identified with “white space” or FET fingers could be lengthened to increase current gain to improve performance. These cells and corresponding transistors were fed to EinsTuner in order to improve slack while remaining within the boundary of the cells and using only the available white space. This methodology, IBM layout aware tuning (LAT), enabled increased performance to critical macro designs with minimal disruption to the physical layout.

In addition, the IBM migration-assist shape-handling (MASH) capability was used to optimize layouts in order to impose stress on FET channels to improve their performance. The result was auto-optimization balancing power/performance requirements.

Table 2 Features of the IBM CMOS SOI technology.

n-FET gate L_{poly}	40 nm	
p-FET gate L_{poly}	35 nm	
Gate oxide	1.12 nm and 2.35 nm	
Metal layers	*Pitch*	*Thickness*
M_1	200 nm	135 nm
M_2	200 nm	175 nm
M_3	200 nm	175 nm
M_4	200 nm	175 nm
M_5	400 nm	350 nm
M_6	400 nm	350 nm
M_7	800 nm	570 nm
M_8	800 nm	570 nm
M_9	1.6 μm	1.2 μm
M_{10}	1.6 μm	1.2 μm
V_{dd} (logic supply)	1.15 V	
V_{cs} (array supply)	1.15 V	
V_{io} (I/O supply)	1.20 V	

3. Fabrication technology

The POWER6 microprocessor is fabricated in a state-of-the-art IBM 65-nm proprietary twin-well complementary metal-oxide semiconductor (CMOS) SOI technology [1]. This technology features ten levels of copper interconnect, in addition to other features that are given in **Table 2**.

The wiring planes consist of four levels of single-width, single-thickness wiring, two levels of 2X width and thickness,[3] two levels of 4X width and thickness, and two levels of 8X width and thickness (widths and thicknesses are shown in Table 2 and a wafer cross-section is shown in **Figure 3**). The technology features several innovative aspects, including a low-*k* dielectric on the lower eight planes. The upper two planes of wiring use standard fluorinated tetraethylorthosilicate (FTEOS) material. Controlled collapse chip connect (C4) solder-ball technology is used for chip-to-package interconnections.

Electronic fuses (eFUSEs) are used to reduce the cost of manufacturing and testing and to allow faster eFUSE

[3] *2X wire width and thickness* indicates that a given wiring plane contains wires twice as wide and thick as a minimal wire in a given technology. A 2X wire has smaller resistance but larger parallel plate capacitance to adjacent wires. A greater-than 1X wire is typically used for timing-critical paths.

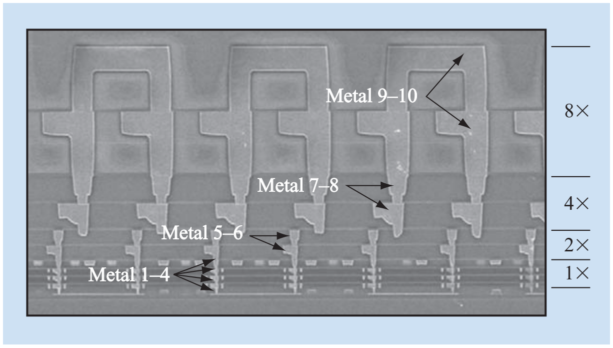

Figure 3

A 65-nm wafer cross-section showing metal layers.

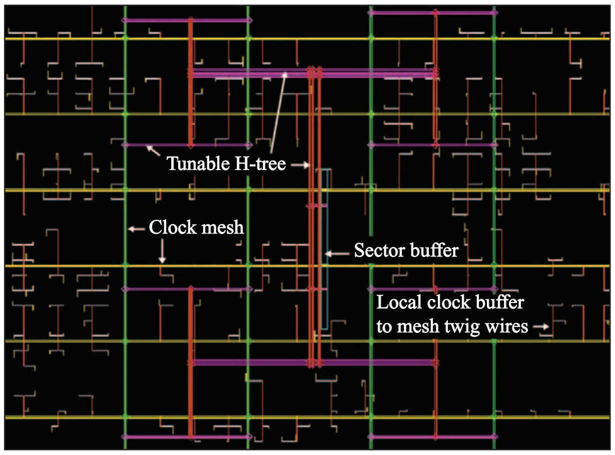

Figure 4

Illustration of the clock network from the output of the sector buffers, through the tunable H-trees to the mesh, and down to the local clock buffer inputs via twig wires (located throughout the chip).

personalization. A precision p+ polysilicon resistor is used in place of the buried resistor (BR) diffusion resistor. In addition, tensile and compressive strain liners are used to improve the respective drive strengths of n-FETs and p-FETs. Automated routines were developed to customize the strain liner layers after final tuning and timing optimization for best performance, even in areas of reusable standard-cell circuits.

As in previous product generations, dual gate oxide thicknesses are used: 1.12 nm for high-performance devices and 2.35-nm oxide for low-leakage devices and for devices that are subjected to higher voltages. An accumulation-mode thick oxide decoupling capacitor structure is used to minimize current leakage. In addition to low-VT, regular-VT, and high-VT transistors, a superhigh-VT (SHVT) device was added for use in arrays.

Significant effort and attention were given to yield and design-for-manufacturability considerations. For example, much effort was devoted to maximizing contact redundancy. For robust power distribution, triple-width metal with 2 × 2 arrays of vias was used for power interconnect. Also, wherever space permitted, additional redundant signal vias were added in post-design processing (**Table 3**).

There are two significant differences from the earlier POWER4 and POWER5 microprocessor [2] technology, which were used in the POWER6 processor 65-nm technology [1] in order to reduce cost and increase yields. First, a fixed 250-nm coarse grid was imposed on polysilicon gates in all logic and array circuits in order to improve channel-length control and device tracking. Second, the local tungsten interconnect layer, which was used in previous technologies, was not used so that yield would be improved and process complexity reduced. Extensive use was made of high-VT devices in non-performance-critical logic, with added use of programmable longer-channel-length devices for additional power savings.

4. Clock design

Global clock distribution design

The POWER6 microprocessor global clock distribution methodology was derived from the one used on previous pSeries server chips [2]. The design contains 621 clock buffers, providing clock signals at three different frequencies to five separate meshes across the chip. Two full-frequency core meshes and one half-frequency nest mesh cover the entire chip and share a single phase-locked loop (PLL). Two memory controller meshes are driven from a second PLL. A single distribution was created for each frequency, resulting in three independent distributions. Each distribution consists of a multistage buffered tree sourced from a PLL, which drives a set of large sector buffers composed of three inverter stages. Each sector buffer drives a tunable H-tree, which is the last stage of tree wiring. The H-tree connects at many approximately evenly spaced points across the clock mesh. The local clock buffers (LCBs) in each macro are wired using a bottom-up method to the desired chip-level clock mesh. **Figure 4** illustrates the sector buffer to LCB network. The increasing size and complexity of the POWER processor chips require improved methods for design and optimization of the clock distributions.

Methodology and improvements

The multistage buffered trees between the PLL and sector buffers consist of seven logic inversions (via inverters) for the memory controller or nine inversions for the core and nest distributions. All were designed as symmetrically as possible in order to maintain low skew at each buffer stage and at the inputs to the sector buffers, even with power supply voltage and process variations. In previous chips, the large trees were designed and tuned manually, with repeated simulations. Because of the high clock frequencies required, technology scaling, increasing variability concerns, and an overall increase in complexity, a more sophisticated approach was taken in the POWER6 microprocessor. Initially, a cross-section of the tree network was designed and carefully optimized. Derivative-free tuning [5] was used to optimize the wire lengths, wire widths, and buffer sizes at every stage in the simulated tree views. Transition times, overshoots, undershoots, skew, duty cycle, power, and sensitivity to process and voltage variations could all be included in the objective function. After the cross-section was optimized, it was copied and expanded into a network covering the entire chip. A tool specifically designed for this purpose was used to move the optimized tree network into a physical layout while adhering to the POWER processor image and to other blockages. The entire buffered tree was then resimulated with the necessary manual wire adjustments being made as the chip design converged.

The final H-tree networks driven by the sector buffers were tuned to account for asymmetric wiring, non-ideal sector buffer placements, and varying device and wire loads seen on the mesh. As in the previous POWER processors, a tuning method [6] was used to adjust the wire widths in order to meet a slew specification required by the LCBs. The tuning also reduced skew across the mesh and subsequently at the inputs to the LCBs. New to the POWER6 microprocessor is the ability to tune sector buffer sizes in addition to H-tree wires. The first inverter stage of the sector buffers remained constant, while the second and third stages were adjusted to change the output drive strength. This tuning allowed additional refinement, more consistent slew at sector buffers driving different loads, skew reduction across the clock meshes, and an overall decrease in power.

The POWER6 microprocessor design improves upon the previous methodology used to connect the LCBs to the large clock meshes. A virtual mesh with reserved wiring tracks was implemented so that a customized routing tool created a twig path from the LCB input pins up to the clock mesh, following the reservation of the virtual mesh. This modification allowed more flexibility in the locations of the clock routes while still maintaining the clock distribution requirements such as slew and skew. The main benefits of the new method occurred at the unit integration level, thereby reducing wire congestion for easier signal routing and allowing more flexibility in floor planning. The change also resulted in smaller wire lengths on the higher-resistance metal layers and, therefore, reduced the load driven by the clock distributions and clock power.

Table 3 POWER6 redundant via statistics.

Level	*Total power + signal*	*Signal only*	*Redundant signal*	*Non-redundant signal*
V1	96.1%	92.0%	4,801,740	55,218,200
V2	95.6%	89.3%	2,847,220	23,752,700
V3	99.3%	97.3%	290,683	10,439,300
V4	97.1%	80.5%	464,152	1,918,450
V5	98.8%	92.2%	134,461	1,593,210
V6	96.0%	78.6%	152,179	559,335
V7	96.6%	82.4%	103,208	482,031
V8	79.9%	75.1%	39,876	119,941
V9	69.4%	81.3%	17,267	74,847

Physical design and results
As mentioned above, there are three distinct clock distributions on the chip, two synchronous distributions driven by a single PLL and one asynchronous distribution driven by a second PLL. The first synchronous signal is distributed to the two cores running at a frequency of >4 GHz. The majority of this distribution was designed as a hierarchical element inside the core (see Figure 1), with only the first four buffer stages outside the core. In its entirety, the synchronous distribution to a single core consists of 44 tree buffers that drive 88 sector buffers with a slew target of 44 ps at the LCBs. In order to further reduce skew in the clock design, wires in the same stage are shorted across the core at selected locations.

The second synchronous distribution supplies a clock signal that runs at half the core frequency of the rest of the chip, which is called the *nest*. The nest distribution is the largest on the chip, with 200 sector buffers that drive 3,174 locations on the mesh. It also contains the only continuous clock mesh. The LCBs connected to this mesh run at half the core frequency, which allows the distribution to be tuned to a higher slew target. Using clock control signals, the nest frequency can also be supplied to the two memory controller meshes. The sector buffers for the memory controller are discussed in further detail below.

Prior to the fourth buffer stage of the synchronous distributions to the core and nest, there exists a programmable-delay buffer. This buffer is non-inverting and can be used to introduce a controlled delay into the trees. Because the two core and nest clock meshes are not

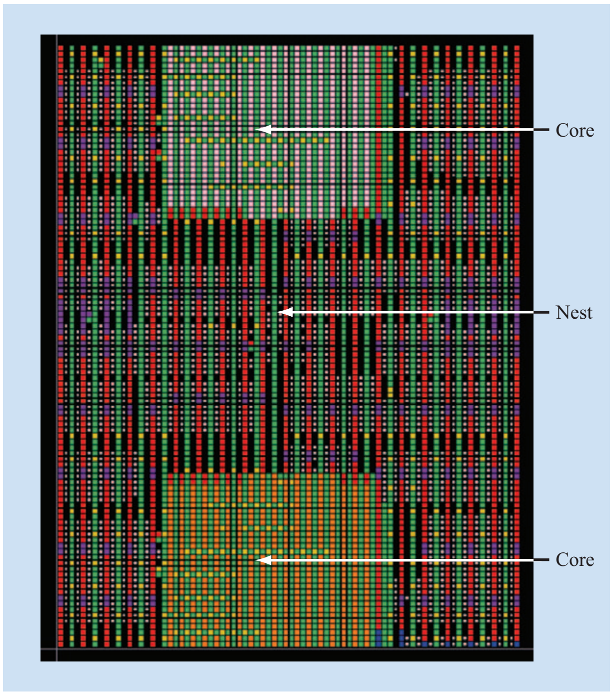

Figure 5

The POWER6 microprocessor C4 power pin footprint with color-coded voltages and white circles representing the signal I/O location.

shorted together, there could be potential static clock skews across the meshes. A high-resolution, high-linearity, programmable-delay clock buffer was designed to alleviate these static clock skews. The programmable-delay buffers used in the POWER6 microprocessor have a range of 40 ps and a step value of 2 ps. The delay control settings may be determined empirically or from measurement circuits such as "skitters" [7]. The programmable-delay buffer may also be used to shut off the clocks to cores in partially good chips for additional power savings.

The memory controller blocks are driven by the asynchronous distribution sourced from a second PLL. This distribution contains 44 total buffers and runs at a frequency of 3.2 GHz. The sector buffers in this distribution are more complex than those in the synchronous distributions. Most of the chip runs on a single voltage plane, but the memory controller blocks run on a different voltage plane at a higher level. The memory controller sector buffers allow for voltage-level switching, thereby increasing the voltage level of the output signal from that of the input signal. Each sector buffer also contains a multiplexer that is used to choose between the nest synchronous signal and the asynchronous signal. This is accomplished by providing control signals and a nest tree input to the sector buffers on the memory controller distribution. The multiplexer allows the memory controller blocks to run at the nest frequency for testing and other purposes.

Because of the size of the clock networks, simulating a single distribution took a few days. To reduce simulation and turnaround time for tuning, the distributions were simulated in multiple sections. Once the individual sections were tuned, they were combined and simulated, verifying that areas near section boundaries were not violating any clock specifications such as local skew. Finally, each distribution was successfully simulated in its entirety, thus ensuring a robust clock design.

5. Integration

Chip floorplan

The early phase of the chip floor-planning process was dominated by the effort to balance the area and wiring resources across the cores, large units, buses, and the chip C4 footprint. Figure 1 shows the location of the major blocks. The two cores are placed on the north and south edge of the chip and four 2M level 2 (L2) cache arrays are located in the four corners of the chip. The core load/store units are aligned with the L2 controllers in order to minimize the L2 cache access path. The cores operate on a 1:1 clock, the nest operates on 2:1, and the memory controller operates asynchronously at ~1.5:1.0. The off-chip drivers and receivers are distributed in horizontal bands across the chip. The 1,953 signal I/Os on the chip support five symmetric multiprocessor (SMP) fabric[4] buses, including three on-node (X, Y, Z) and two off-node (A, B) level 3 (L3) buses, in addition to a memory bus and a GX bus.[5] In order to control the impedance of the C4-to-driver or receiver wires, the distance was limited to 800 μm. The chip has 5,449 power I/Os that were divided into six voltage domains: V_{dd}, the default for logic; V_{cs}, for static RAMs (SRAMs); V_{io}, for I/O drivers and receivers; V_{d0}, for core 0; V_{d1}, for core 1; and V_{sb}, for standby power. Because the cores operate at highest frequency and have the highest power density, the area under the cores is void of signal C4s and has the highest concentration of power C4s. **Figure 5** shows the chip C4 footprint with color-coded voltages and white circles representing the signal I/O locations.

The multiple voltage domains presented new challenges to the physical design process:

- Power distribution regions were defined with power overlap shapes that were recognized by the power bus generation tools.

[4]*Fabric buses* allow all nodes on a bus to connect to one another. These are used in multicore designs and multichip designs to connect all the nodes.
[5]*GX bus* is used to connect to an I/O drawer on the system.

- Voltage translation circuits were required for signal-crossing the voltage domains. The voltage translators require access to multiple supply voltages. Voltage attributes were added to all macro I/Os to facilitate checking for translation consistency. A new tool, an IBM power rail checker (VIPER), was developed as a postprocessor for the VHDL [Very-high-speed integrated circuit (VHSIC) Hardware Description Language] compiler to check that all signals crossing a voltage region have the appropriate translators. Global EinsCheck, another new IBM tool used for electrical checking, was developed to verify electrical connections and design rules across domain crossings.
- The automated buffer insertion tool, an IBM buffering tool (AddBuf) had to be updated to recognize the power regions, place buffers in the appropriate regions, and add the appropriate power connectivity.
- Power clamps had to be added to smaller voltage regions to provide electrostatic discharge (ESD) protection. For small regions, the intrinsic capacitance is too small to protect the circuits from an ESD event. ESD is the phenomena that occurs when static electricity discharges into a circuit (usually from human handling) and can severely damage the circuitry.

In order to achieve frequencies greater than 4 GHz and minimize latencies, use of the higher-level faster wiring planes had to be optimized. Priority was given to the clock, power, and C4-to-I/O wires on the upper planes; anything remaining was made available for signal routing. A tagging process was developed that allowed for specifying wire codes and a preferred wiring layer range on a per-net basis. Initially, the net tags were set on the basis of length, and then they were updated on the basis of timing feedback. A full range of wire width and spaces were defined for performance and power optimization, as shown in Table 2.

The buses were wire-coded early in the design process. The wire code information was stored in a directory of files that could be updated and then reapplied to the netlist with each new logic drop. After a floorplan that could be wired was established, future wire code updates had to be reviewed by the integration team.

A fabric bus plan was developed that allocates the wiring resource that connects the core and units across the chip. The bus plan was also used as an input to the buffer grid generation. Buffer slots were allocated over units.

Compared with the previous POWER processor design, major enhancements were made to the automated buffer insertion tool, AddBuf. The tool now supports automatic latch insertion. This feature was used during early iterations of the floorplan. Latches were auto-inserted in the floorplan so that timing could proceed in parallel with adding the required latches to the chip VHDL. The tool was updated to place buffers automatically in legal physical locations, thereby allowing most of the buffers to be added without manual intervention. The chip has ~500,000 buffers, with the majority being automatically placed. The tool also supports automatic placement of spare buffers. The global phase of the router was used to guide the buffer placement on the chip. This minimized the impact of the buffers on the detailed router. The process generated good results except for the cases in which buffer slot availability was limited. Feedback from the global router was also used to evaluate wiring congestion and guide floorplan moves.

New steps added to the physical design flow

- Auto-place new clock and data staging macros. To facilitate the synchronous control of all the chip clock control blocks, a ten-stage clock control pipeline was distributed across the chip. An automated process was developed to connect the clock controls to the appropriate stage. After new logic was placed, the placement information was used to reassign the clock control connectivity and back-annotate the VHDL.
- Review tag updates and apply updated tags to netlist.
- Reassignment of scan clocks on the basis of region.
- Run global router and pass results to AddBuff.
- Add flues. *Flues* are via stacks that go from the macro pin up the higher wiring planes and help minimize driver source impedance and reduce signal EM problems.

On the basis of the timing results, fails were fixed with buffer updates, wide code updates, or logic changes.

Routing

The chip, core, and unit routes were done with Cadence CCT routing tool. The routing process was set up to support the wire classes defined in Table 2. The target was to have 100% via redundancy on the vias for the 1X layers (see Table 3 for results).

6. POWER6 power distribution design and analysis

The POWER6 processor chip power distribution was partitioned into multiple domains to enable dynamic voltage throttling [8]. The use of multiple voltage domains on the POWER6 processor chip drove a unique set of challenges for the power distribution design and specification. Two voltage rails for array macros were required to power the SRAM cells and the supporting logic (V_{cs}), while logic and interface were powered by the logic power domain (V_{dd}). V_{cs} isolation was required since the SRAM cells have functionality issues at low-voltage operation. In addition to array macros, additional power rails were required for off-chip circuitry (V_{io}, V_{sb}) to provide a constant common interface level for intrachip communication. The memory controller units were also powered by V_{io} because this function must interface asynchronously with off-chip memory logic. The PLL function was also powered by the V_{io} domain to allow constant operating voltage.

The design of the power grid to support the various voltage domains across the chip required a good understanding of the placement options needed by the design group to achieve optimal floor-planning capability as well as the load current demands for each placed object. In order to eliminate the need of providing more than two voltage rails and ground to any portion of the chip, a floorplan constraint was placed to isolate off-chip circuitry in common regions throughout the chip. Since arrays required more floor-planning flexibility, a global power grid containing V_{cs} was used outside the I/O areas. Within the I/O areas, which included the memory controllers, a V_{io} grid was included. V_{dd} and GND (ground) were interleaved with these two rails throughout the chip.

A coplanar design approach in which signals and power are interleaved on all levels of metal was used, while power delivery to the chip was provided by solder-ball connections. In order to establish the appropriate ratio of power and GND to all portions of the chip, current demands for each object had to be known. This information was provided on a macro basis by analytical techniques (see Section 7) that incorporated active and leakage power estimates as a function of macro FET widths and latch counts. Using this load data, a power grid could be specified and evaluated using an IBM power grid analysis tool called the *Austin linear simulator flow* (ASF) [9].

ASF provides complete power grid analysis capability from concept phase evaluation to final verification. The basic engine within ASF is a shapes processor and extractor that formulates a circuit matrix directly for the resistive power grid. This eliminates the need of an intermediate SPICE (simulation program with integration emphasis) file. Load information, i.e., macro placements and power estimates, is sourced into the tool via a power map file. A linear simulator is employed to evaluate the circuit for voltage decrease and EM from C4 to the lowest level of metal. For concept phase evaluation, ASF has a built-in power router that allows the user to define and evaluate power grid options quickly. This capability facilitates sensitivity analysis of various power grid options to determine the optimal distribution of power stripes for each level of metal and for each power rail. With this approach, power pitch and widths were specified for each level of metal. In areas where dual-voltage domains existed, V_{io} or V_{cs} stripes were substituted for V_{dd} on the basis of the current splits and internal resistance (IR) drop requirements for each rail. Because all currents sink to ground, GND distribution occupied 50% of the power grid real estate (space), while the other 50% was divided between V_{dd}, V_{io}, and V_{cs} on the basis of load requirements.

In addition to specifying the on-chip power distribution, C4 placements played a key role in minimizing voltage drop. Because of high current demands in certain areas of the chip and the need to supply dual voltages within certain regions, C4 placements became contingent on the floorplan. Thus, periodic evaluation of the chip power grid was performed to ensure adequate C4 placements with respect to floorplan changes. This evaluation loop was performed several times before the package design freeze date.

Final power grid verification included the voltage drop and EM of the actual power grid layout with more accurate macro power estimates. ASF accepts power grid data in the form of a PowGeo file, which defines the power grid physically. In order to translate designs in Cadence to a PowGeo, an intermediate VLSI (very-large-scale integration) instance model (VIM) database was built for each unit in the design for the level being evaluated. Because of complexity and data volume issues associated with macro layouts, a Niagara (IBM shapes infrastructure tool) function was adopted to convert this Cadence data to an IBM netlist format VIM for the macro power grid only. This process, called *macro power grid abstract* (MPA), an IBM macro power grid abstractor tool, resulted in significant data reduction. VIM flattening and translation to the PowGeo was done within the IBM ChipBench framework. In addition, early analytical macro power estimates were replaced with a transistor-based analysis tool known within IBM as *CPAM* (common power analysis methodology) [10]. CPAM incorporates linearized device models to facilitate fast simulation of the macro netlist for a set of input vectors. The output from CPAM is a set of maximum average currents at each contact area (CA) via location within the macro and is stored in the IBM DB2* database. ASF reads this data for all macros in the design and attaches the CA currents to the power grid via placement information in the design power map. The power map is a physical description of the power grid. A flow diagram of the ASF verification process is shown in **Figure 6**.

To improve turnaround time and reduce memory requirement, final verification of the power distribution

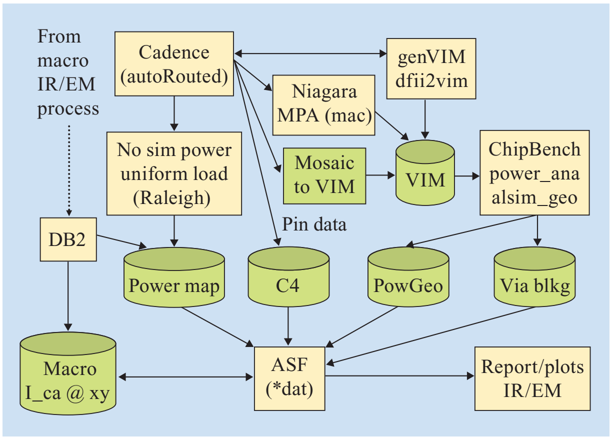

Figure 6

ALSIM flow (ASF) process used to validate power grid integrity. (IR: internal resistance; EM: electromigration.)

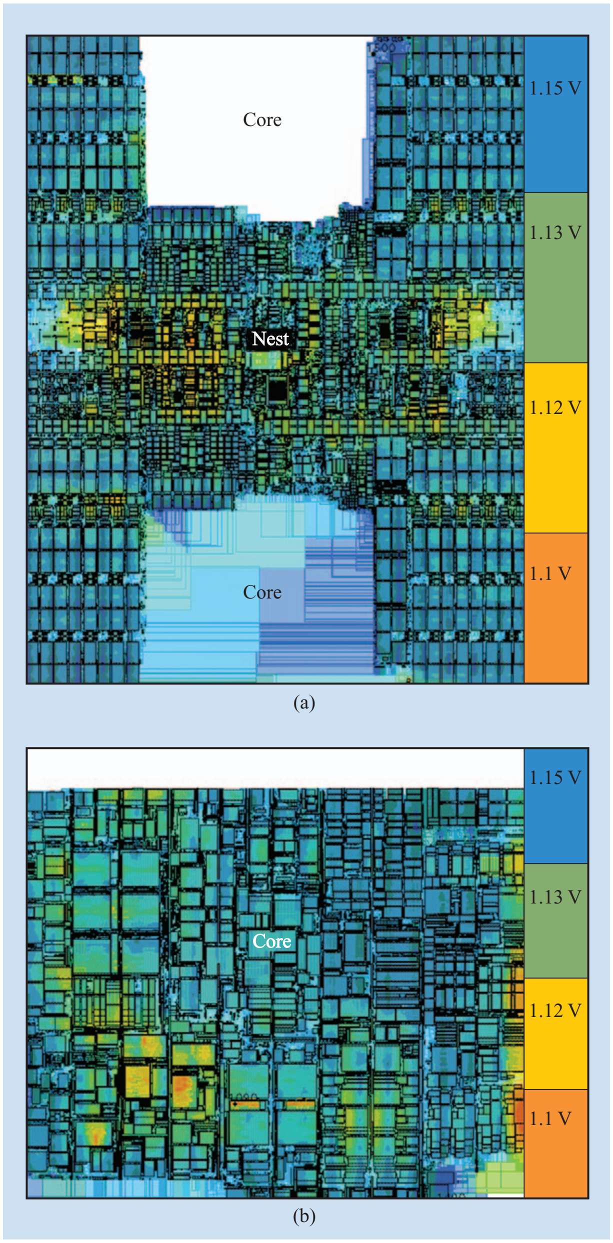

Figure 7

V_{dd} voltage contour maps for (a) nest (or non-core) and (b) core areas.

was performed on the nest and core areas independently. Since the two cores and associated C4 placements are mirrored and are exact copies, only one core had to be evaluated. Such an approach allowed nest and core evaluations to proceed in parallel, which further reduces turnaround time.

Contour maps generated from ASF for the V_{dd} rail are shown in **Figure 7** for both the core and the nest. These contour maps provide visual feedback of voltage drop issues associated with power grid implementation problems or clustering of "hot" macros. If hot spots violated the voltage drop specification, additional power shapes were placed in available signal tracks to bolster the grid in the offending areas. In some cases, the macro power estimate was reviewed to determine whether high-power activity was real or could be limited through circuit design modifications.

POWER6 physical design and chip integration methodology

The POWER6 dual-core microprocessor chip was designed in four levels of hierarchy from an integration perspective. This is similar to other server microprocessor chips designed in the IBM Systems and Technology Group (STG) [2, 3, 11]. The macro, unit, core, and chip levels were designed concurrently with extensive track-sharing across all levels of metal. Examples for each level of hierarchy include an adder for a macro, a fixed-point unit for a unit, an entire microprocessor core for the core level, and the nest and multiple core subsystems for the chip. Teams were located across the globe, so design logistics had to be optimized. Each unit team consists of logic architects, logic designers, logic verification personnel, circuit designers, a timing take-down leader, and an integrator. Circuit designers generally are responsible for the macro level of hierarchy. Macros contain transistors and two to five levels of metal. Integrators are used to place the macro blocks and wire the level of hierarchy (unit, core, chip) to which they were assigned.

Getting data management and physical design processing correct was a significant issue throughout the design cycle in order to minimize schedules. Physical design content varied throughout the development process. During high-level design (HLD), virtual integration techniques were developed to support rapid 3- to 4-day timing design turnaround times to the logic team. Virtual techniques include rudimentary abstracts, timing-rule estimates, non-legalized buffer placements, virtual latching, flattening unit hierarchies, and routing of wires using the global router (i.e., no detailed wiring), as well as limited macro wiring contracts. A description of the timing take-down strategy is described in the section "Chip timing closure."

To support the rapid update of logic deliverables into the physical and timing reduction flows, an internal library management system was used and is summarized here but described in more detail in Section 9. Having a global team across several time zones that spanned 7 hours placed a number of requirements on the design data and tool infrastructure:

- Design intellectual property (IP) had to be shared among other microprocessors. Design IP includes physical blocks/sub-blocks and VHDL. This IP was shared throughout the hierarchy among four other chips (primarily at the macro and unit levels of hierarchy). Interlocking levels or configurations of data between chips were successfully transferred to prevent deletion of data when one chip no longer required a particular configuration of blocks.
- An audit subsystem (see Section 9) was required to run nightly on the master data repositories to yield some rudimentary verification of methodology and to compare last-changed checks against tool execution date-and-time stamps.
- The "unit" level of hierarchy usually is the long path in the schedule for releasing the chip data to mask manufacturing. To process logic drops efficiently in physical design, the unit processed logic drops asynchronously from other units for the majority of the design cycle. This was done to allow the unit to proceed rapidly and independently. Some units could take drops every 7 to 10 days, while others might be on a 10- to 14-day drop cycle. Discontinuities occurred at times, but all were usually resolved within the next logic drop. It was only toward the end of implementation that synchronous updates were used for final verification, a point at which all data had to match up exactly. Several weeks of efficiency were gained using this procedure.
- Tools and data had to be accessed locally to reduce network latency. An extensive shadowing system was used to keep sites synchronous. If a site was taken offline (planned or inadvertently), other sites could continue to work.
- All physical design data had versions with check-in and checkout capability, and multiple users had to be able to create a checkout in parallel with others. "Branching" in the data manager (DM) was a quick way to make editable views for chip analysis work.
- Multiple data levels were used in the data repository for each logic drop to deliver data to various verification teams. It was common to use a separate level for noise analysis, design for test, estimated timing, extracted timing, physical design, and physical design verification. The version and interlock mechanisms in a library management system allowed each group to see a static set of data for analysis.

Several enhancements were made to the integration HLD and implementation methodology in order to support the POWER6 microprocessor. Some changes also control a design process across disciplines. Complex metal blockage (keep-outs) contracts can arise across any layer of hierarchy. To foster better communication and blockage management across the hierarchy, an abstract generator (ABG), the IBM hierarchy blockage management tool, was developed to interlock both the cover and the abstract of any delivery of the hierarchy. At the macro–unit interface, ABG was used to formalize the pin, block size, and blockage map agreement between circuit designer and integrator. At other levels of the hierarchy, ABG solidified this wiring agreement among multiple integrators. Core and chip nonnegotiable content (power, clocking, and C4 wiring) was passed down through the hierarchy as a system of parent covers using a tool not described here. This section presents both the HLD and the ABG techniques in depth.

High-level design

In previous projects, the physical design was locked into decisions the logic team had made without detailed input about the physical implementation. For this project, we were able to provide feedback about the physical implementation to the logic team earlier than ever before and build floor-planning information into the physical microarchitecture and latch point decisions. This was paramount to support the ultrahigh frequency targets for the POWER6 microprocessor. To do this, methods were required to deal with an incomplete logical design and incomplete macro definitions. This is an early physical implementation of HLD, and in some cases, virtual methods were used to facilitate this. Virtual methods are most often used early in HLD and allow a designer to "cheat" on the current physical design in order to be able to complete something quickly. If the design cannot pass timing with these “cheats,” then they are not necessary. As the design progressed, fewer virtual methods were used and the design moved closer to its release to manufacturing.

Early in the design process, the macros are often far from complete. There are several procedures that can be used to give us a reasonable approximation of the macro. Logic designers are given tools to approximate the size of a macro on the basis of its cone of logic[6]; this is also used to generate an estimated timing rule for the macro. Given this size and experience with other similar logic, the macro abstract can be defined and wiring resources can be allocated over the macro using ABG and by assigning pin locations.

Early in HLD, the netlist was flattened through units to the macros. Using this flattened netlist, rough areas were allocated to a unit and its macros were placed in this area. This step was undertaken before all the data stacks had been implemented. In regions where there were known logic deficiencies, placement area would be reserved and wiring blockages would be created to simulate the wiring resource this logic would need. The global router was then used to test the feasibility of the overall chip design. Many what-if scenarios could be quickly tested, and a viable floorplan with placements and major bus routing decisions could be solidified. When a floorplan was selected, the units could begin their implementation. The units were given an outline that is based on the placement of their macros and a wiring contract that is based on the congestion map, thereby providing the unit with resource to wire its nets and the top-level resource to wire its nets through the unit. This early flat environment was invaluable to ensuring that good early decisions were made in macro placement and bus routing.

With the contracts in place, units could begin to work more autonomously and begin to attack the specific problems in their units. Initially, the I/Os of the units were forced to be snapped to the driver or sink[7] of the macro, allowing the top level to manage the path to the pin. This reduced churn associated with keeping the edge pins of units in the correct place while the unit macros and the units themselves were still in placement churn. When this placement churn began to settle, the pins of the unit were snapped to the boundary. Top-level routing was used to locate a position on the edge of the unit by examining where the route left the unit.

Buffering during HLD has various modes beginning with virtual buffering and ending with complete buffer placement. Early in the design, it is not worth the effort to completely buffer the design with accurate buffer placements. If early virtual buffering with ideal buffer locations shows timing issues, these must be fed back to the logic team before moving to more detailed buffering. Several virtual buffering methods are available for use. The timer can be programmed to assume ideal buffers during RC calculations for the net. Alternatively, the buffering tool can be set to ignore blockages and place buffers at the ideal location on the net. During this procedure, the global routes of the nets can also be used to drive the location of the buffers. As the design improves, detailed placement and analysis of the buffering solution becomes worthwhile.

Routing during HLD also follows this approach of fast or virtual early runs to provide feasibility feedback, with more detailed routing runs being performed only as the initially uncovered problems are fixed. One of the most significant problems with routing early in HLD is access to pins. Initially, not much time is spent building the cell abstracts or ensuring that top-level infrastructure and macro placements are all in sync, which causes pins to be difficult or impossible to route to. Virtual buffering also causes issues with accessing pins. A method was implemented of cutting holes around a pin and on all layers above the pin in order to allow the router to be able to access the pins. At first, only global routing runs are undertaken to check the design. These run fast and give us views of congestion that can be used to analyze the current floorplan. These global routes are also used to drive pin placements and buffering decisions, and they can provide input to the timer to get congestion impacts into the timing runs.

The primary reason to be involved in this HLD is to provide timing feedback. There are several modes in which timing can be run (see the section on chip timing closure, below). Integrators trade off turnaround time versus accuracy in order to provide timing results for their unit. The early HLD timing analysis quickly identifies problem areas that the integrator and logic team can attack. In previous designs, this timing feedback would not have been available until much later in the design cycle, when logic and partitioning changes are expensive. Most fixes were done through placement and wiring resource (i.e., width and layer assignments) changes. The new HLD methods allowed us to make this feedback available much earlier and to identify these logic and partitioning changes. This enabled us to reduce the time it took to physically design this chip.

Abstract generator

Unlike the physical design methodologies used in previous POWER microprocessors, the methodology used in the POWER6 microprocessor employed a single application, the IBM abstract generator (ABG), for the creation and management of detailed blockage contracts at all levels of hierarchy: macro, unit, core, and chip.

[6]A *cone of logic* can be defined as many logical inputs being compared and resulting in a few outputs.

[7]A *driver* is the gate output and a *sink* is a gate input. Pins were snapped on to these connections.

Furthermore, ABG was designed to support all design phases. For example, early in the macro design phase, ABG supported the use of abstracts as the source for pin locations. Later in the design phase, macro layouts became the final source for pin locations. Such flexibility ultimately allowed for highly concurrent, hierarchical design phases.

One challenge in designing a single application to support blockage contract management at all levels of the hierarchy is the inherently different design flows used. For example, blockage information is traditionally transferred from the macro level to the unit level. This bottom-up approach makes it difficult to communicate stable wiring resource needs to upper levels of the hierarchy. Conversely, chip, core, and unit blockage information is traditionally transferred to lower levels of the hierarchy. This top-down approach makes it difficult to communicate stable wiring resource needs to lower levels of the hierarchy. Therefore, to support a highly concurrent hierarchical design approach, it was necessary for ABG to support bottom-up and top-down design flows simultaneously. In order to do this, features were incorporated into ABG that were previously available only in distinctly separate applications. First, traditional wiring contract management concepts were introduced. However, the method by which this was accomplished was unique. Rather than clutter the ABG graphical user interface with a plethora of special-purpose widgets, three new contract cell views were introduced to guide ABG: *cellName*_mine, *cellName*_yours, and *cellName*_nobody. Their respective purposes were to convey to ABG the wiring resource reservations required by the child cell view, the parent cell view, as well as any exclusive wiring resource reservations that neither the child nor the parent cell view could use. These cell views contained shapes drawn by the designer, which represented the literal wiring resource reservations required. In order of precedence, these wiring resource reservations will hereafter be referred to generically and conceptually as *mine*, *yours*, and *nobody* reservations.

In combination with the graphical contract cell views previously described, another cell view was introduced, *cellName*_image, hereafter known as the *image file*. This text-based cell view defined a special data structure that conveyed a host of advanced functionality to ABG. In addition to keeping complexity out of the graphical user interface, another key feature of the image file was repeatability. Because the controls and tacit assumptions were maintained in the image file, they were persistent and reproducible regardless of who ran the application. A few of the key blockage modeling features and techniques that were controlled by the image file are now described.

One key component of any wiring contract management system is how the area around a pin shape is modeled. Traditionally, pin regions present a unique problem because they represent an area that is "owned" by both the child and its parent. This shared wiring resource is ultimately represented as a lack of blockage around each pin (in the abstract) or pin location (in the cover). As a result, steps must be taken to ensure that design-rule correctness is maintained in pin regions. The traditional approach, which dates back to the POWER4 microprocessor design project, was also employed in the design of the POWER6 microprocessor. In order to maintain design rule checking (DRC)[8] routes, child metal shapes around the area of each pin were modeled as net shapes. This conveyed their locations to the router so that it could, in turn, avoid common DRC violations such as notching errors.

Other pin modeling features that were based on learning experience from previous projects were incorporated into ABG and deployed for use in the POWER6 microprocessor. One unique innovation was the ability of the designer to control pin cutouts (blockage shape erasure) on the metal layer directly above the pin. Furthermore, these cutouts could be confined to pins with particular geometries (lengths and widths). In addition, the designer could also enable a feature that would insert via obstructions on the layer directly above the pin. These features could be used selectively during times in which unique design situations would not always guarantee pin access to the router.

Modeling of clock pin regions in ABG was given additional attention. Because of the critical nature of the clock mesh, connections to clock pins must be made as directly as possible and with minimal wire jogging. To that end, ABG automatically created an implicit *yours* reservation at each clock pin location. These reservations had the following characteristics: The blockage shape was created on the same metal layer as the clock pin, the blockage shape width was that of the clock pin, and finally, the blockage shape length spanned the entire breadth of the place-and-route boundary and was oriented in the preferred wiring direction of the metal layer in question. These features were customizable using the image file, including whether the clock reservation scheme was enabled for the questionable cell.

Another key blockage modeling feature that was deployed and heavily used in the POWER6 microprocessor design was an edge reservation scheme. Quite often, macros placed their primary input and output pins at or near their place-and-route boundary. This was especially true for our synthesized macros. The density of these pin shapes could lead to unique routing challenges and congestion at macro boundaries. To address this issue, an edge reservation scheme could be

[8] *Design rule checking* implies that the physical design shapes meet spacing criteria specified in the technology manual.

enabled and customized using the image file. Through this track-based reservation scheme, the designer could define the number of edge tracks (from the place-and-route boundary inward toward the cell center), their boundary locations (north, south, east, or west), as well as their reservation type, namely *mine*, *yours*, or *nobody*.

Similar to the graphical *cellName*_mine, *cellName*_yours, and *cellName*_nobody contract cell views, the image file could also be used to create *mine*, *yours*, and *nobody* reservations. The image file implementation was track based, which lent itself to defining periodic, pattern-based wiring reservations. For example, macros were typically granted exclusive use of the first four metal layers, but only a small percentage (if any) of metal layer five (perhaps in one out of every four or five wiring tracks). The image file made defining these requirements trivial.

The image file also supported a method for defining region-based track reservations. This scheme allowed the designer to define one or more bounding boxes that marked out regions within the overall place-and-route boundary where special consideration was required. The capabilities inherent to defining the overall place-and-route boundary region were also available to these subregions. For example, the track-by-track periodicity, metal layer, and track ownership (*mine*, *yours*, or *nobody*) were all customizable settings definable in the image file.

Early resolution of the bottom-up and the top-down specification of the wiring contract naturally fell to the level of hierarchy that was completed first, which was almost always the macro level. Because the wiring contract was specified separately from the design data, the next highest hierarchical level ultimately had control of its preferred definition of the wiring contract. This method was a unique departure from previous POWER microprocessor projects, in which control was largely driven by design data contents, which were extremely variable, as well as by graphical user interface settings, which were easily forgotten. Ultimately, this scheme of separation created the right environment for hierarchical wiring contract maintenance because it forced a continuous, open line of communication between designers working at all levels of hierarchy.

In keeping with the inherent give-and-take nature of hierarchical wiring contract maintenance, another key ABG innovation that was new to the POWER6 microprocessor was the use of a distinct order of precedence for the *mine*, *yours*, and *nobody* reservation scheme (**Figure 8**). Because a *yours* reservation could reclaim wiring resource that was previously defined in a *mine* reservation, the POWER6 microprocessor design team was provided with a simple, yet powerful, method for defining and manipulating wiring contracts. This unique ABG capability, along with those previously described, enabled highly concurrent design phases for the entire POWER6 microprocessor design team, which ultimately contributed to the success of the project.

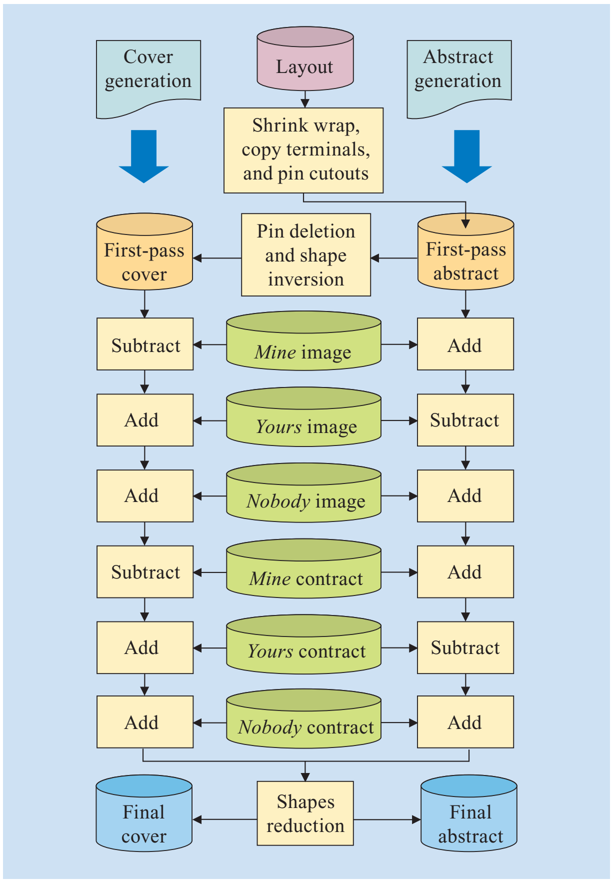

Figure 8

Bottom-up abstract generator (ABG) flow depicting abstract and cover generation, as well as the order of precedence of image-file and contract cell view processing. Primary input data: rose cylinder; optional, user-generated input data: teal cylinders; intermediate output data: orange cylinders; internal ABG processing steps: rectangles; final output data: blue cylinders.

Chip analysis closure

Chip timing closure

Timing of the POWER6 chip brought with it many challenges. With any challenge comes advancement in process and technique. The cross-site macro and global timing teams at IBM provided common timing tools and a methodology solution that was state of the art. This resulted in an overall timing methodology with roots in previous POWER processor designs [2].

The POWER6 microprocessor timing closure effort required rapid iteration and parallel execution at all levels of the hierarchical design, which included custom macro, RLM, unit, core, nest, and top-level chip timing. In order to enable this concurrent timing closure, the development and fine-tuning throughout the entire project of a timing assertion methodology were necessary to provide accurate boundary conditions that would be transferred down from the top level to the lower levels of the hierarchy. These conditions were in the form of timing contracts of arrival and required arrival times, input slews, and capacitance loads, which were key inputs to unit-level timing as well as to EinsTLT and logic synthesis processes. In addition, slack apportionment techniques were used in some areas of the nest design to divide slack targets among macros, which enabled faster overall timing closure for the logic in redesign.

The levels of timing accuracy throughout the POWER6 chip timing closure naturally followed the progression of the design. Timing during the early HLD phase consisted of zero-wire-delay analysis to assess logic partitioning issues when floorplans were not yet available. The next timing modes used as the design progressed included cycle reach, virtual buffering, virtual latching at the unit, core, nest, and chip levels as floorplans became more stable, while buffering and staging latches were not yet in place. Following was Steiner-based analysis, with per-net wire code and wire layer-use information as input to the timing run. Global routes, later followed by detailed routes in the timing run, were also supported in modes of operation in which the routing information was read in part of the design (VIM) netlist. The final and most accurate level of timing was a fully three-dimensional extracted parasitics-based timing analysis for all levels of the design. Throughout the design cycle, as changes were made, the timing analysis of functional units was often in a fallback, less-accurate mode until the physical design was able to catch up to the next level of accuracy. With such fine granularity available in the timing methodology, the POWER6 microprocessor design had the most accurate timing information available at any given time, which led to fewer timing surprises along the way.

At the global chip level, the hierarchical design approach led to a hierarchical timing approach and often presented the chip timing analysis with a mix of accuracy levels available at a given time. Since the different areas of the design do not progress equally through the analysis modes described earlier, a method was needed to use the best timing information available at all levels. Some functional units were farther along in the design cycle with extracted parasitic representations of global wire or logic macros, while other pieces of the design were still in a Steiner estimate or schematics-based mode of timing. The global timing process, which executed under IBM ChipBench (IBM floorplanner) invoking the EinsTimer static timing tool, supported a true mix of parasitics in the design across all levels of the hierarchy. The wiring and electrical subsystems of the tool would effectively stitch together the most accurate parasitic information available for each piece of the hierarchical source or sink connection before network reduction and analysis. This enabled true concurrent timing closure in a hierarchical environment.

In previous designs such as the POWER5 microprocessor, the timing closure required multiple analyses at each timing corner to close on functional and scan-related timings. For the POWER6 microprocessor design, through enhanced circuit modeling and clock phase definitions, only a single timing run was needed at each timing corner to gain visibility into functional timing, system scan, and tester scan timings. This single analysis also covered multiple voltage domains and multiple design frequencies throughout the chip. In addition, implemented on the POWER6 processor was the use of clock pulsed latches, which enabled increased frequency and power savings in the design. Through the clever definition of nearly 70 clock phases, the only analyses that were necessary include the basic fast and slow chip timing plus a noise impact on timing analysis (coupled noise analysis) to cover all system modes of operation.

Coupled noise impact on timing techniques, which are used to determine the delay degradation due to capacitive coupling and simultaneous switching effects, was greatly improved over that of previous POWER processors. Previous analysis involved a wire-to-wire coupling estimate (*k*-factor) approach that translated into a capacitive uplift on a wire, thus translating into an increased delay effect. Folding in a true noise analysis invoking IBM 3DNoise (IBM noise analysis tool) [12] in EinsTimer using the IBM EinsSI (IBM coupling impact on timing subsystem) enabled calculation of the actual noise introduced on a victim net from its aggressors and subsequently determined the change in delay on the wire and slew on the net. This resulted in more accurate analysis than in previous designs, which was warranted at these tighter cycle times and wire-delay-per-path restrictions in the POWER6 processor.

In order to achieve rapid chip timing closure in the POWER6 microprocessor, automation and consistency of input design data and output reports were key objectives. At all levels of global timing from unit, core, nest, and chip, the timing team implemented a hands-off approach to timing setup and execution. In previous designs, input data specification processes for design netlists (VIMs), macro timing rules [dependency check macros (DCMs)], and extracted parasitic information [parasitic data models (PDMs)] had to be done manually, but in the POWER6 processor, these are fully automated.

Using the same chip-level IBM CAD library[9] (ICL) control file, which contains the design bill of materials to ultimately build, check, and "tape out" the chip, enabled the automatic derivation of inputs to the timing run, with all the information needed to perform the analysis without designer intervention. This allowed consistency and accuracy of results during the closure process. The introduction of the IBM view late timing report (VLTR) into the POWER6 processor flow enabled storage and retrieval of report information in a DB2 database. VLTR was used in the chip timing closure process to assign path ownership and document timing path solutions and to generate timing statistics and histograms to show timing take-down progress throughout the design. VLTR was also used as an analysis tool with the ability to invoke the EinsTLT for a path-based transistor-level timing expansion of an abstracted path from a global timing run using its input assertions. Consistency ultimately enabled fewer false iterations throughout the timing process.

Consistency of timing results in the POWER6 processor design was equally important. Traditionally, RLM logic was represented by standard cell gate-level delay/slew equations, which differed from the more accurate transistor-level analysis employed in custom logic. To reduce this variability, each custom macro or RLM was represented by a transistor-level abstract timing model during the chip-level analysis. The benefit of equal treatment in analysis ultimately translates into easier hardware timing debug, knowing that the overall design has a more deterministic path ordering.

Because the design of the POWER6 processor was so complex, consisting of millions of signal nets, more than 500,000 buffers for slew closure, more than 45 million latches, 1,165 unique design macros with 11,998 instantiations, and more than 50.4 million late-mode timing checks, automation, consistency, and accuracy of analysis were key to achieving a high-frequency design.

Figure 9 shows the resulting late slack histogram for the chip timing closure process in the POWER6 processor design at the target frequency for reference. (*Slack* is the amount by which a timing check passes or fails. Negative slack indicates a failing check.)

Timing accuracy improvements

Timing accuracy was improved for clock uncertainty and array timing. Furthermore, transistor-level timing noise analysis was added to minimize problems when a design is implemented in hardware.

Clock uncertainty is a significant component of the cycle at this frequency. This uncertainty was broken into its various components and modeled separately to improve the accuracy of the timing analysis.

[9]The *IBM CAD library* is a design data manager primarily written as a data manager to Cadence design environments.

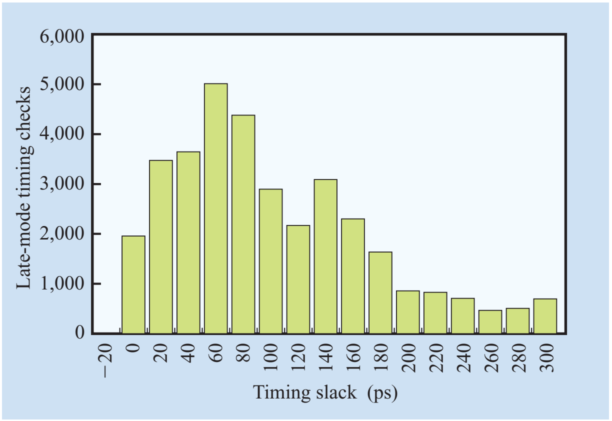

Figure 9

POWER6 microprocessor timing histogram depicting distribution of timing slacks for the frequency goals of the chip.

Furthermore, a delay modifier was applied to the early and late switching times for each circuit in order to account for increasing delay variability due to process variation.

The embedded arrays historically have been in the cycle-limiting paths for processor designs. Many of the array macros were completely timed using the EinsTLT tool. This was used instead of relying on PowerSpice (IBM SPICE tool) simulation results that must come from a manually generated subset of the design to obtain the timing models for EinsTimer to use further up in the design hierarchy.

To meet the cycle-time requirements in the critical paths around the embedded SRAMs, several aggressive design techniques were used:

- Logic functions that would normally be performed in RLMs were implemented as custom logic inside the custom array macros.
- Self-timed programmable pulse generators were used for clock generation.
- Extensive use of cycle stealing through these paths reduced the impact of clock arrival time variation on the delays.

In the past, we could use PowerSpice simulation on a subset of the design to determine whether the array macros would meet timing requirements; however, this technique is labor intensive and error prone. Significant amounts of logic added to these designs made this approach impractical. This was addressed by improving the static timing analysis tool (EinsTLT) so that it could handle macros that contain arrays.

Several problems had to be addressed to make this possible:

1. A significant number of timing paths and checks are associated with timing all the cells in an array. Timing all the paths into and out of every cell can result in prohibitively long runtimes for timing analysis at the macro and chip level, as well as large timing rules.
2. Read/write bitlines are bidirectional. The read and write operations had to be analyzed separately.
3. EinsTLT contains basic support for domino circuits and clock gating and shaping. The custom circuits that are in the embedded array macros use complex domino and clock gating and shaping circuits. EinsTLT had to be enhanced to have robust support for domino and clock gating and shaping circuits in order to improve its checking and accuracy.

The number of timing paths and checks that were required for the arrays was reduced by exploiting the fact that the layout for arrays tends to be very regular. The earliest and latest switching times for the wordlines and bit columns tend to be at the edges of a physical subarray. For example, consider the cells with the earliest and latest switching times to and from the cells as being the outside of a donut, and the switching times at the pins of the cells are within the donut hole and fall between those around its edge.

The *donut-hole* cells were retained in the simulation model so that real wire loading and capacitance values could be used. EinsTLT was stopped from timing into or out of the donut-hole cells by preventing the model from trying to create timing information for these cells. We then post-processed the resulting timing data to ensure that the cells were timed with the worst-case timings at their wordline and bitline pins.

POWER6 functional noise analysis

To ensure functional operation of the interconnect infrastructure on the POWER6 processor in the presence of various noise sources, a combination of analyses were performed in conjunction with the application of an upgraded server group noise methodology to predict and bound potential problems at various phases in the design process.

Functional problems can occur when the composite noise signature is greater than the circuit noise margin. Predicting the actual noise signature is difficult and can only be bounded in practical terms. The noise signature has traditionally been thought of as a combination of coupling effects, common mode, and differential mode power supply variations. A recent study by Deutsch et al. [13] demonstrated that differential mode noise adds inversely to the common mode and coupling effects. The study also concluded that with even moderate power supply decoupling, the common mode assumption for extraction is adequate and accurate. Therefore, the differential power supply effect is neglected when considering functional problems, since this noise source is always less than the circuit noise margin. In addition to dynamic effects, leakage biasing in 65 nm [1] technology had to be considered. It was observed through an internal study that skewed beta-ratio circuits could result in significant biasing effects due to subthreshold leakage. In addition, gate leakage could create voltage drops between source and sink locations for high-fanout net topologies, which effectively reduced the circuit noise margin. These dc biases were treated as additional noise sources in the POWER6 processor noise methodology.

Although circuit functionality problems could occur when the total noise is greater than the noise margin, stage-to-stage propagation could result in evanescent behavior of the noise. This effect is already taken into account using the IBM macro noise tool IBMmlsa (IBM macro-level signal analysis) [14]. At the global level, multiple buffer stages that provide logic connections between pins are prevalent throughout the design. In order to predict the noise propagation effect on these circuits, a new noise abstract model was formulated to calculate the buffer output noise as a function of input noise and output load capacitance. This formulation was incorporated into the IBM noise simulation tool, 3DNoise. In addition to predicting noise propagation through buffers, functional failure is determined by noise levels at latches or the number of buffer stages exceeding the noise limit. The latch failure criteria are accounted for by rerunning IBMmlsa for noise failure at input pins using a noise-assertion criterion. The noise assertion is the dc level and pulse amplitude/width determined by 3DNoise for every input pin of the macro.

The global noise methodology used in the POWER6 processor is shown in **Figure 10**. The IBM 3DX global extraction tool operates in GL1 (IBM graphic language; a shape format), a translated data format from the Cadence database, and outputs frequency-dependent resistance, inductance, and capacitance (RLC) data for all nets as required [14]. 3DX output is a PDM. Noise simulations are performed for every net in the design using 3DNoise, which incorporates the $R(f)L(f)C$ data as a synthesized circuit for evaluation. In addition to the noise calculation, timing windows, which represent the switching interval for each aggressor and sink sensitivity interval, were generated by the global timing tool, EinsTimer. 3DNoise determines the composite noise amplitude and pulse width on the basis of the aggressor window overlap within the receiver sensitivity window. In order to determine noise failure, noise margin calculations were performed for all macro inputs using IBMmlsa, which also provided the impedance data for all macro outputs. Both subthreshold and gate leakage biasing were calculated at the macro level within IBMmlsa and fed forward to 3DNoise for evaluation via the noise abstract. IBMmlsa also provided special-purpose noise abstract for buffers to facilitate noise propagation within 3DNoise. This was obtained by calculating an output noise table as a function of output load capacitance and input noise level for each buffer in the design.

The POWER6 processor design was unique in the sense that chip wire routing never occurred over the core areas. This allowed the functional noise verification process to be implemented on the core and nest independently. However, because of the high level of wiring interaction between lower-level units and either the chip or the core, a flat evaluation was required for each design. IBM GL1 preprocessing code was used to merge unit and chip (or core) GL1 as a preparation step to run 3DX.

Noise problems resulting from the net-based evaluation flow were reevaluated through the use of either the buffer noise propagation procedure or the noise assertion at the macro input. If failing nets were easy to fix by either rebuffering or moving wires, buffer noise propagation required input noise information for the victim driver and noise results at each of the victim sinks. In addition, downstream buffer information was also gathered and fed into 3DNoise for evaluation using the modified buffer noise abstracts. 3DNoise would calculate both incident and propagated noise and compare the total noise with the circuit noise margin for each stage. If more than two successive stages failed this criterion, the failing net was fixed by rebuffering or moving wires. In the situation in which a noise fail occurred at the input to a macro, the macro designer reran IBMmlsa with noise assertion to observe propagation within the macro. If noise at latch inputs was excessive, the global net had to be fixed. Although these two processes were available to reduce the need for a net-based analysis, in some cases fixes were easily identified, which minimized the implementation of these additional procedures.

Signal electromigration

Unilateral current flow can lead to mass transport caused by electromigration (EM). Interconnects (wires and vias) degrade by EM effects. EM reduces interconnect reliability by shortening the time to failure (TTF). Several models for EM exist, e.g., see Black's equation [15]. Basically, the TTF is inversely proportional to a power of the dc current density j_{dc} and exponentially proportional to the inverse of the temperature T: TTF $\sim \exp(c/T)/j_{dc}{}^n$. As shown by this equation, EM quickly gets worse at higher temperatures. On the power distribution

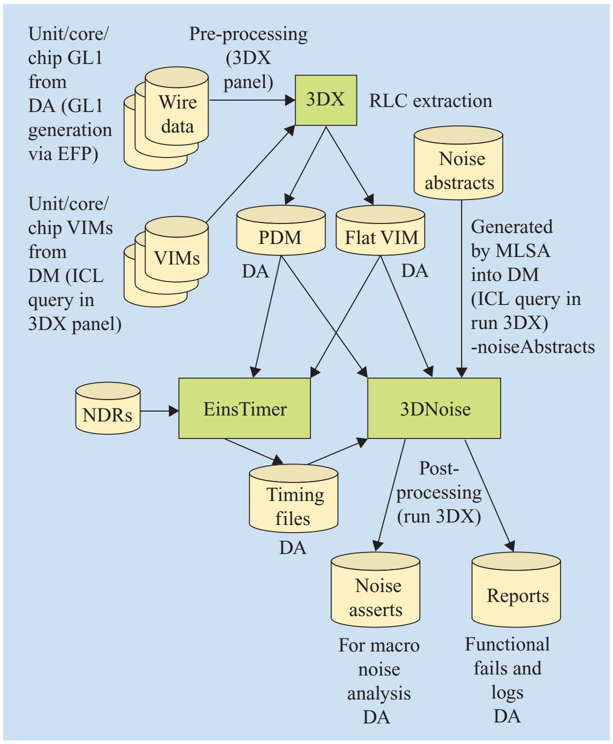
Unit/core/ chip GL1 from DA (GL1 generation via EFP)
Wire data
Pre-processing (3DX panel)
3DX
RLC extraction
Noise abstracts
Unit/core/ chip VIMs from DM (ICL query in 3DX panel)
VIMs
PDM
DA
Flat VIM
DA
Generated by MLSA into DM (ICL query in run 3DX) -noiseAbstracts
NDRs
EinsTimer
3DNoise
Timing files
DA
Post-processing (run 3DX)
Noise asserts
For macro noise analysis DA
Reports
Functional fails and logs DA

Figure 10

Global functional noise flow. (RLC: resistance, inductance, and capacitance; VIM: VLSI instance model; ICL: IBM CAD Library; PDM: parasitic data model; MLSA: macro-level signal analysis.)

nets with strict unilateral currents, EM was calculated and checked in previous projects and it was also checked in the POWER6 microprocessor project. This section describes the secondary effect on signal lines by temperature increases on signal lines and how it was measured in the POWER6 microprocessor.

Global signal EM

Global signal lines between macros carry a bidirectional current because of charging and discharging the parasitic interconnect and transistor gate capacitances. Even with the gate leakage, the dc component is negligible for EM of a signal interconnect for the 65-nm CMOS process. The bidirectional current causes self heating (or Joule heating) of the interconnect because of the electrical power dissipation due to the resistance of the interconnect. Because of thermal conductivity of the material, self heating causes a temperature increase not only in the interconnect itself but also in the neighboring circuitry.

Self heating not only induces EM on neighboring interconnect with dc current components such as the power distribution and local interconnects between

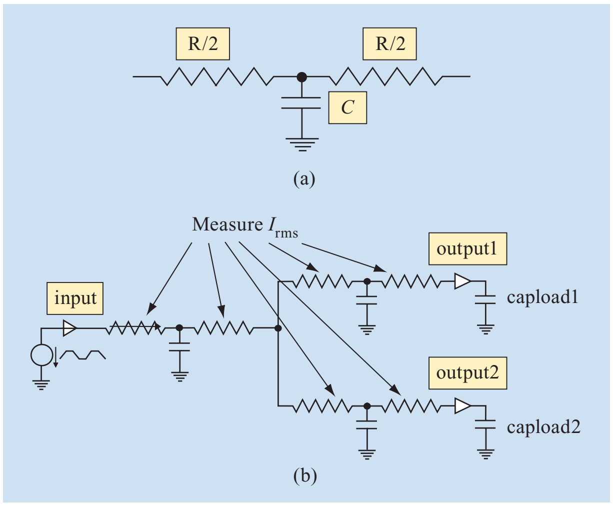

Figure 11

(a) A T model consisting of two resistors of equal value with capacitance in the middle is commonly used to model a wire. (b) An electromigration measurement scheme. Currents are measured at each node in the RC network in which a single gate drives two gates.

transistor drains and sources, but also causes a temperature gradient along the interconnect that might lead to additional failures in the interconnect. In order to control the self heating of the interconnects, design guidelines were set to limit the temperature increase caused by self heating to $\Delta T_{\max}$. Because the heat flow in the silicon and interconnect material and the chip cooling are well known, the maximum allowable electrical power dissipation $P_{\max}$ per volume can be calculated from $\Delta T_{\max}$. Likewise, the maximum allowed $I_{\text{eff,max}}$ can be determined for each interconnect layer (of known thickness) depending on the electrical effective width w_{eff}. I_{eff} is also known as I_{rms} ($I_{\text{root-mean-square}}$) because of the formula to calculate:

$$I_{\text{eff}} = I_{\text{rms}} = \sqrt{(switchingFactor/t_{\text{cycle}} * \int i^2(t)\mathrm{d}(t))}. \quad (1)$$

Design guidelines tabulate the maximum allowed $I_{\text{rms,max}}$ values per layer with a dependency formula on the width.

Each global signal interconnect was analyzed to find $I_{\text{rms,max}}$ violations by performing automated SPICE analyses. 3DX extraction provided a PDM database, C++ classes, and methods to access the parasitic data and netlist information. With these methods, a SPICE netlist was generated from the PDM for each signal net. The netlist contained an RC representation of the global signal interconnect. Each wire segment and via was represented by a T model consisting of the elements R/2, C, and R/2 [**Figure 11(a)**].

From a database, the layer, width, and coordinates were retrieved for each capacitance of a wire segment or via. Industry-standard SPICE-measured I_{rms} and I_{average} statements, together with a normalized ratio $I_{\text{rms}}/I_{\text{rms,max}}$, were added for each resistor in the netlist. The $I_{\text{rms,max}}$ was calculated from the width of the wire segment or via when the netlist was completed. Capacitance loads representing macro or book input pin capacitances and a voltage source with a trapezoidal pulse representing a driving circuit with a certain voltage swing, cycle time, switching factor, and slew were also added to the netlist. Switching factors were also determined from extracting the timing phase of the signal from EinsTimer [**Figure 11(b)**]. The electrical values such as voltage swing, driver slews, resistors, and capacitances were based on fast process parameters representing the worst case for I_{rms}.

The netlist contained only linear elements, which made the automated SPICE analysis very rapid. Nets that had a total net capacitance below a project-dependent defined threshold value were ignored because they could not cause excessive self heating. In order to eliminate the dc components of the currents, multiple cycles were analyzed and the I_{rms} measurements were read out in the last cycle. The results from the SPICE measure statements were automatically inspected for a ratio >1.

Violations with a ratio >1 were reported along with I_{rms}, $I_{\text{rms,max}}$, net name, cycle time, switching factor, driver slew, voltage swing, layer, width, and coordinates. A marker file to locate the failing segments easily in a layout editor was also produced. For each failing net, an I_{rms} report of all wire segments and vias was written. Waveforms and results for each wire segment could also be provided.

Several options were available to eliminate existing I_{rms} violations. The individual options listed below could be combined:

- Widen the metal or add more vias.
- Choose a thicker metal layer.
- Split the net to reduce capacitive load and buffer the net.

Signal electromigration analysis within macro designs

Given the vast circuit styles and wiring topological complexities, the signal EM analysis within the macro required not only circuit simulation to find problematic failure connections but also analysis of circuit-logic state. This analysis sensitizes the circuit configuration to enable the maximum charge or discharge current so that failing wiring connections or vias can be located. With these requirements, the IBM static timing tool, EinsTLT, would integrate signal EM analysis within the circuit design analyses. Also within the macro-based analysis application, the transistor interconnections that assemble logical gates can have both unidirectional and bidirectional currents that must both be measured against technology guidelines.

In order to allow for unique identification of the interconnect wiring and vias, the macro parasitic extraction tool, IBM Erie (macro extraction tool), includes specific information for the analysis of signal EM and provides information so that the EM analysis failures could be located within the physical layouts to correct the design. The *x*- and *y*-axes coordinates, the metal or via layer name, and the effective resistance of each individual wire segment and via were to be included so that the failing structure could be identified during the analysis. As in static timing analysis, the EinsTLT application utilized similar methods in developing each network structure that was to be passed to the embedded circuit simulator. In this case, current measurements were returned on wiring and via components for comparison to the technology guidelines. During a timing analysis accomplished by EinsTLT, a conditioned cycle time of the circuit under test must be applied by accounting for cycle times during current measurements so that signal EM can be determined. Additionally, I_{dc} and I_{rms} currents are also dependent on the switching behaviors of the signals propagating through the circuits, which are also included during the analysis with the results reduced by a switching approximation for each signal.

7. Power and performance analysis

Introduction

Analysis of power for the POWER6 chip presented several new challenges. In addition to chip size and complexity, multiple system applications of the chip existed with different power constraints and performance requirements. Therefore, power and performance had to be modeled with sensitivities to voltage, frequency, and temperature while preserving variations in the manufacturing process. In addition, the multiple power supply voltages drove the need to estimate not only total power but also power by voltage rail so that system power supply requirements could be determined early in the chip design cycle.

Leakage power of the 65-nm devices was a much more significant fraction of total chip power than in previous technologies. Circuit designs had to include additional devices at differing threshold voltages to permit leakage/performance optimization. Consequently, acquisition of design data that affects leakage power had to be organized by device type, and the leakage models had to be implemented for each unique type. Design techniques for mitigating ac power dissipation included low-power latch operating modes. Estimates were required for each unique mode as well as counts for latches not only by power level but by operating mode.

Power estimation methodology

Power dissipation was modeled in two components: ac power and dc power.

AC power

AC power was estimated prior to completion of schematics or VHDL by using a latch count and area-based approach. The power dissipation that results from simply clocking the latches of a typical macro is a significant portion of the total macro power.

Rather than require a circuit simulation to be performed on each macro, simulations were done on each latch type to determine its power dissipation at a set frequency and voltage with its data inputs held constant. Latch counts for each latch type were estimated for each macro in the design. Actual latch counts were extracted from design data later in the design cycle. The power dissipation that resulted from clocking the macro with all its inputs static could then be accurately estimated by simply multiplying its latch counts by the power per latch. The resulting *latch power* was then scaled to application voltage and frequency. Additional macro power dissipation resulted when its inputs were not static.

Simulation results from previous chip designs were analyzed to determine the incremental macro power that was a function of input data switching activity. This incremental power (data power) was found to correlate well to macro area. While data power from an individual macro could vary significantly from what would be predicted on the basis of its area, the total data power of a large number of macros could be accurately estimated from its area and application conditions. Additionally, the data power component of a macro with typical input switching activity is relatively small compared with its latch power. The result of this work was a closed-form model of the ac power of each macro as a function of its latch count, area, and application voltage and frequency.

DC power

DC power is the component of total power dissipation that is independent of frequency. It consists of parasitic device currents (subthreshold and gate tunneling current) and resistive power dissipation from I/O termination or other resistive networks across the voltage rails.

Parasitic device currents (leakage)

Subthreshold current is also known as *channel leakage* or I_{off}. I_{off} is a function of W_{eff}, L_{poly} (or device length), V_{dd}, and T_j.

I_{off} equations provide a normalized A/m gate width that is used with gate width data to calculate channel leakage at any application voltage, temperature, or manufacturing process line center.

In addition to subthreshold or channel leakage current, devices suffer from gate tunneling current. Unlike I_{off}, the current path for I_{gate} is from the device gate to its drain or source and is significant when the device is on. For I_{gate} calculations, 50% of the devices are assumed to be on or to have V_{dd} as their V_{GS} or V_{GD}. Fifty percent of the gates are assumed to be in the off state ($V_{\text{GS}} = 0$ V, $V_{\text{GD}} = V_{\text{dd}}$).

I/O macros employed resistive termination. These were simply modeled as resistors with power proportional to the square of the voltage across them.

Performance estimation methodology

Both chip $f_{\max}$ and power are strong, nonlinear functions of V_{dd}. In addition to estimating power at a specified voltage, for the POWER6 processor, it became necessary to estimate power at a specified performance level ($f_{\max}$). Performance estimates were based on the assumption that chip delay paths consist of a silicon delay portion that scales with voltage, temperature, and process just like a process-sensitive ring oscillator (PSRO) and a wire delay portion that is relatively insensitive to V_{dd}.

Simulations were performed on various PSRO circuits, and regression analysis was performed on simulation results to get a closed-form model of PSRO as a function of V_{dd}, T_{j}, process variables, and device type used in the PSRO. The worst-case delay path varies with application conditions. As V_{dd} is raised, the silicon portion of delay paths becomes a smaller fraction of the total delay, while the wire delay portion remains relatively fixed. Therefore, multiple delay paths were modeled that represent the extremes of wire delay versus silicon content for paths dominated by high-VT device and nominal-VT devices. A function to calculate $f_{\max}$ in terms of application conditions and process parameters resulted.

Data acquisition and storage

During the design process, VHDL and schematic data became available for latch count and area estimates. Software was developed to exploit existing tools to extract the design data and populate it into a structured query language (SQL) database. Use of a SQL database permitted use of an "off-the-shelf" spreadsheet and chart-making tools to access the data, create trend graphs, model power, and minimize required tool development resources.

Spreadsheet generation

Spreadsheets are useful for interactive analysis and graphing. However, as model complexity increases, they become error prone and difficult to understand by users other than their author. It is easier to read and understand well-commented procedural code than to examine spreadsheet cells if one is experienced in the procedural language used for the model.

Experience shows that the biggest source of power estimate inaccuracy was frequent mistakes in spreadsheet implementations. Rather than develop code in the spreadsheet itself, procedural code was used to create the spreadsheet. The same code was used to generate spreadsheets for each of the supported processors so that a consistent model was used for every chip power model and maintenance resource was minimized.

A common template spreadsheet was distributed with menus extended to allow generation of project-specific spreadsheets. In addition, the generated spreadsheets did not contain static design data, but they could be refreshed with a simple mouse click so that the latest (or any previous release) design data would be used.

Circuit-limited yield

The closed-form models developed for power and performance estimation made it feasible to perform chip-level Monte Carlo power and performance analysis. Based on manufacturing test data, process parameters were modeled as statistical distributions. Aggregating these results allowed insight into circuit-limited yield (CLY), or wafer yields. Design gate width and latch count data were aggregated by domains. A unique set of voltage, temperature, and device types constituted a leakage domain. AC domains consisted of each unique voltage and asynchronous frequency, latch type, and macro type (RLM versus custom).

Random variables were generated for each simulation sample. From these random variables, variance about median values was calculated for process variables. Additional variation was added to model contribution from unknown sources to ensure that power/performance variability was consistent with measured hardware data.

A graphical user interface (GUI) was developed for the tool to allow specification of either a power or a performance constraint. The CLY tool would initiate a Monte Carlo analysis in which each would perform a search algorithm to determine the maximum V_{dd} that could be used within a power constraint. This V_{dd} value was then used to determine the $f_{\max}$ given the process and random variables of a specific sample. The $f_{\max}$ and the values of all varying parameters were saved so that results and random data used could be later plotted as histograms and cumulative probability distributions. Similarly, an $f_{\max}$ constraint could be specified and the tool would find the minimum V_{dd} required to meet that constraint. Power was then calculated at the $f_{\max}$ constraint and associated data was saved for analysis.

Model-to-hardware correlation

The power, performance, and CLY estimation mechanisms used in the POWER6 microprocessor showed good correlation to actual hardware results. In fact, correlation was significantly improved compared with previous programs in which SPICE simulation-based methods of power estimation, which require significant engineering resources and computation time, were used. AC power as measured in W/GHz*V^2 correlated to estimates within 10% across a range of operating conditions [**Figure 12(a)**]. Similarly, dc power and $f_{\max}$ showed excellent correlation across the process [**Figures 12(b)** and **12(c)**]. These correlations enabled accurate prediction of product frequency and ship yields; this was significant to the timely delivery of systems meeting performance goals.

8. RLM design

As described above, the hierarchical design of the chip created units that must be designed to fit into the chip. The unit design team must understand function, area, timing, and power constraints and decide the best mix of RLMs and custom macros. Control logic is likely to be changed as design features are added or removed. As unit and chip verification progresses, logic should be implemented as an RLM and remanded to tools for implementation. If related logic is properly grouped into reasonably sized RLMs, the implementation tools do about as well as a human designer would, although only if provided good input, for example, on timing constraints, RLM pin placement, and reasonable amounts of logic in each cone. This division of the unit into RLMs improves turnaround time because each RLM can be built on a collection of background processors. Generally, the entire nest unit could be run in a few days, while other core unit RLMs may take as long as a few weeks, for example, because of interaction with routing, noise, and EM. The cost of creating and maintaining this hierarchy must be paid upfront.

The unit design teams were staffed with logic designers that owned the function of the RLMs. RLM builders managed the RLMs through the implementation process. Unit timing teams managed the assertion and validation process, and a unit integrator owned and managed the interaction of all the unit RLMs and custom macros as well as the integration into the chip. Often, a single person could serve many of these roles. The most successful units managed their designs with budgets for area, timing, and power that did not vary much for each implementation. This stability led to the discovery of design-level problems sooner and resulted in much more predictable implementation runs.

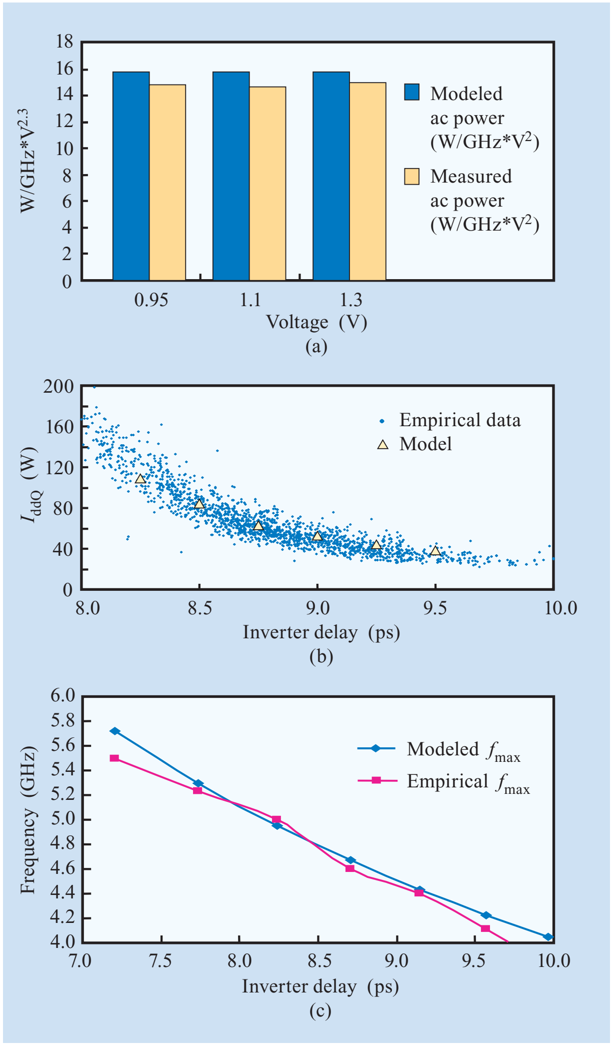

Figure 12

POWER6 microprocessor (a) actual (modeled) vs. measured (hardware) ac power; (b) I_{ddQ} vs. inverter delay from model and hardware [(a) and (b) republished with permission from [16]; ©2007 IEEE]; (c) $f_{\max}$ vs. inverter delay for model and hardware.

RLM implementation

Once the unit structure was designed, each RLM had to be implemented. IBM tools that worked in a unified framework, such as ChipBench or IBM infrastructure (Nutshell), were used, which allowed incremental timing analysis (EinsTimer), while the design was being elaborated and optimized using the IBM BooleDozer* application and IBM placement-driven synthesis (PDS). A standard sequence of heuristic and metric-based optimizations was developed and tuned for a particular technology at a particular cycle time and was expected to work well for a very high percentage (e.g., 90%) of the RLMs it was given. This tuning was generally in the form of setting parameters (or *knobs*) to activate and deactivate specific functions and algorithms, selecting the proper technology-standard cells for each RLM on the basis of its cycle-time target (nest cycle time is twice that of the core), and setting thresholds and cost multipliers to influence the optimizations. Unique technology and chip constraints drove the development of new code and enhancement of base system code. The resistance of the wires did not scale down as fast as the silicon delay improved, so RC was a significant challenge.

The flow used for IBM server processors differed from the PDS flow used by ASIC (application-specific integrated circuit) chips under the IBM flow manager tool (The Guide). The chip hierarchy generally created smaller RLMs that in turn allowed higher effort on each RLM without creating unusually long running jobs. The processor flow runs from the high-level register transfer language (RTL) and includes the technology-independent synthesis (using the BooleDozer application) flowing directly into the PDS and hence was called the IBM PDSRTL. The PDSRTL follows the typical sequence of heuristics used in a PDS flow:

1. Read the input RTL design, delete unconnected logic, and analyze and remove redundant logic.
2. Restructure the design in order to reduce logic complexity. This is one of the major set parameters provided to the RLM builder that affects the results. They range from "do nothing" to completely throwing away the RTL constructs by creating a sum-of-products implementation and then factoring that out to implement the cones of logic.
3. Map to technology standard cells. Perform initial optimizations to bring the logic cones into compliance with technology constraints (e.g., fanout optimization and repower for loads).
4. Perform an initial placement of all the logic to minimize projected total wire length (no timing considered). This ignores clocking and scan constraints. It allows the use of a Steiner-based router that is quite fast and yet differentiates individual wires on the basis of their connection to other logic and RLM I/Os.
5. Begin timing and electrical correction by cloning and buffering logic and restructuring logic cones on the basis of timing.
6. Insert the clocking network for the RLM. The assumption is that the latches have been distributed and are now "near" the logic they interact with.
7. Timing and area are optimized. A significant amount of CPU (central processing unit) time is spent here as various options on restructuring the logic cones, e.g., cloning/repowering/buffering the logic for speed, are evaluated and either selected or rejected. Any constraints implied by the chip circuit lead are met.
8. Electrical violations (e.g., signal slew limits and RC problems) are fixed.

In the POWER6 processor design, the RLM build process began by importing the synthesized, placed VIM into the Cadence framework as an *autoLayout cellview*. Once the data was in the Cadence framework, it is floor-planned using various techniques including instantiating the cover cell into the RLM that contains the blockage definition for restricting the placement of routing data. Another important step in floor planning the RLM is to define the placement and routing grids using the technology-defined requirements. A typical RLM had routing grids defined for the first four layers of metal, with very congested RLMs having the ability to use up to the first six layers after agreements with the unit integrator. For RLM engineering change orders (ECOs), the design was then merged with the previously routed design. It was then taken into the Cadence placement tool, QPlace, to place any new logic EC books where previously there were gate array cells from the initial build. For an initial build, the design was then filled, or refilled for an ECO, with decoupling capacity cells (dcap)[10] and gate array cells for performing any future ECOs.

The next phase of building an RLM is to prepare the design for routing. Large bar pins were used, 0.3 μm $\times$ the height of the macro, to denote the allowable locations on the macro for which the router was able to connect the global clock pin on each LCB. The unused portion of the large bar pins was eliminated from the final design during a post-processing step. Next, the clock nets connecting each LCB to the individual latches were prerouted with a 3X-wide wire adhering to a '+' pattern, with the origin and length of each set of preroutes determined by the placement of the LCB and latches. This step was successful at reducing any clock skew from the LCBs to the latches. In addition, all timing-constrained nets, including the clock nets, would be annotated with either wide wire codes or specific net weights in order to prevent the wires from becoming scenic when routed. The remaining signal pins and internal nets were routed using the Cadence routing tool WRoute. By default, all RLMs are routed with 100% redundant vias, which greatly improved yield results for the design. Fewer than 5% of designs had to use a combination of redundant and

[10]*Decoupling capacity* is used on power grids to minimize voltage fluctuations.

single-cut vias. The fully routed design was then post-processed using various techniques to generate a DRC layout cell view. All floor-planning data was removed from the final layout, as well as any unused portions of the clock preroutes and the large bar pins. This final layout was then taken through the design-checking tools in the same manner as a custom-designed macro. If any macro timing issues were encountered during analysis of the final layout, the design could be updated by defining additional wide wire codes or net weight on the specific nets, which were sent back through WRoute for incremental routing.

From the perspective of the unit design team, the use of these tools for increased productivity was critical as the design was evolving. The weekly churn due to timing problems, noise problems, logic design constraints, electrical checking violations, changes due to verification, addition or removal of function required the RLM implementation flow to provide highly repeatable results that were generally correct by construction. Experience with the tools led to a design of the logic the way the tools expected and the results were good enough to change many macros from custom designs to RLMs. This entire process of using complex tools required a simple interface that could be managed with minimal designer effort. Just as important was the flexibility of the tools to handle specific cases of exception for each macro consistently.

9. Multisite microprocessor auditing and data management methodology

Global project repository

The physical design environment included Cadence integrated with a multitude of point tools. The following auditing and data management methodology was the result of having to find a way to allow multiple sites to work together to produce a design so large that the required people were not available at any one site. The designers were concerned with having the critical data and tools they needed at their site. When one site went down, it was imperative that a significant number of designers at every other site could continue to work. This dictated that the design data and tools that were required by most of the designers at a site had to physically exist at that site. This requirement was fulfilled by locating pieces of the design at the sites that accessed them the most (installing soft links to the data from the other sites) and by shadowing commonly used design data, tools, and control files between the sites. Therefore, each site project repository appeared to be nearly identical.

The logic is stored in a concurrent versions system (CVS) repository specific to one site (other sites link to this repository) and the rest of the design data was stored in ICLs. IBM CAD is an internal electronic DA (EDA) tool that allows the storage of various configurations of the same Cadence library. These libraries were used to manage the content of most circuit and integration data (e.g., Cadence views, timing, and VIMs) for the POWER6 microprocessor and were mastered at the site where most of the designers working on that particular library were located. If particular libraries were read extensively at a site where the master library was not located, those libraries (or a subset of their levels) were shadowed to that site.

Each level in an ICL can contain only one version of a particular *cellView* (each level looks like a standard Cadence library), so levels were used to create different configurations of the data contained within the libraries. These configurations are called **ICL releases** and can be **BOUND** (i.e., contain an ICL level and its dependent libraries as well as their level) or **UNBOUND** (i.e., include a reference to a particular **BOUND ICL release**). All **ICL releases** constitute a complete configuration of a particular ICL level. All **ICL releases** (represented by icl.lib files) are stored under the *iclCommon* level of the library. All design work and auditing was done on the basis of an **ICL release**. This allowed for asynchronous drops for the various libraries so one unit was not forced to wait for another to catch up. **Table 4** shows the required ICL levels for an official POWER6 microprocessor library.

Bound configurations

Each ICL level (except for *iclCommon*) has one **BOUND** icl.lib file kept under the *iclCommon* level of the library. It is forever tied to that particular ICL level, and it must use this level as its name (*level*.lib).

The "librarian" must include the ICL level and *all* of its dependent libraries (and their level) in this file. The path to this **BOUND** icl.lib file can be retrieved by calling the script *eifGetIclLib*. The advantage of this is that if no icl.lib file was stored for a particular level of the library, the librarian can create a standard one and return a path to it.

For example, the standard icl.lib file created for a project circuit library includes only two lines: one to include the shadowed common reference libraries and one to include the current level of the circuit library. For an integration library, the standard icl.lib file includes the reference libraries, the integration library, and all the circuit libraries under the unit for that integration library.

Unbound configurations

An **UNBOUND** configuration is defined by an icl.lib file that does not have an existing ICL level in its name. These **UNBOUND** icl.lib files are used to tell parents of the library (such as the core or chip unit) or users of the library [such as various tool owners: timing, verity, noise,

Table 4 Levels that each POWER6 IBM CAD library should contain.

Level name	*Audit?*	*Content*
workLevel	Yes	Management of content responsibility of the designer. Usually day-to-day work occurs directly in this level. On sites where the master library is not local, working in private libraries is encouraged. This level should be promoted into the *masterLevel* instead of being promoted directly into a release level.
iclCommon	No	This level contains various icl.lib files that are used to control ICL resolution for various timing, noise, logic, and physical configurations.
masterLevel	Yes	The *authTable* entry for this level is set to not editable; this permits asynchronous promotion by the circuit designer and the integrator. The circuit designer promotes into master (using a direct promote from *workLevel*, or a hierarchical copy from a user library) and the integrator promotes out of master into populated numbered levels.

design for testability (DFT), and audit] which level of the library should be used for a particular task.

The unit (typically the unit integrator) is responsible for ensuring that the various **UNBOUND** icl.lib files are up-to-date. After an **UNBOUND** icl.lib file has been created for a unit, it can be updated by using a special tool called *setLibPath* or by checking it out and editing it from the ICL browser.

The audit system gives an example of how these **UNBOUND** configurations can be used. The nightly audit *cronjobs* searches all of the project libraries for **UNBOUND** configurations that begin with "audit_" (e.g., audit_masterdd0, audit_pdd1_1, and audit_p1_ec012). All of these audit_*.lib files will be opened and audit will be run on the release given inside each of them (provided it belongs to the current project).

One course of action strongly encouraged for everyone on the project is to use the *setLibPath* (IBM library-path-setting tool) in the Cadence environment. *SetLibPath* ensures that all designers pick up complete configurations (**BOUND** and **UNBOUND**). This is critical to picking up all appropriate dependencies. Just having an ICL level in your personal icl.lib and then adding dependencies manually could cause one to be out of sync with the rest of the designers using that particular level. This could cause incorrect VIM generation, tool failure, incorrect audit results, or DRC errors, for example.

Auditing

As mentioned in the section on the global project repository, above, auditing is done on a per-ICL release basis. All of our physical design-checking tools (e.g., EinsTLT, verity, DRC, and LVS) are integrated into our audit system. The IBM design auditing tool GPA allows one to see the result of these audit runs. In addition to the audit logs, GPA also requires that a program be run nightly in order to generate information about the Cadence hierarchy (*releaseAudit*). This system is based on the one used for the POWER4 microprocessor. A number of key modifications were required for the POWER6 microprocessor:

- Every time an auditable check is run with the audit option turned on, an audit log is created inside the **ICL release** being audited. This is done by having the auditable check call the *gpaLogSuccess* (IBM gpa subprogram) program.
- One of the innovations added for the POWER6 microprocessor is the P-grade, which conveys when an auditable check was passed with a relaxed set of requirements. While no macro should be finally released with a P-grade, it allows the integration team to determine whether certain minimal checks that are required to allow successful practice integration runs were passed.
- Allow grades for macros to be transferred from one **ICL release** to another during the promotion step used to initialize a new **ICL release** from an old one. This enables checking results to be transferred to another microprocessor project along with the design data for various macros or an entire library. This can be done with the confidence that the audit will pick up any problems introduced by different project-specific requirements.
- Creation of an IBM library view query tool that is composed of scripts that are used to gather the file paths of different data types for a complete configuration. These include *eifGenPath* (returns the search path for an **ICL release** for a specified *viewType*), *eifGetIclLib* (returns the official icl.lib file for the **ICL release** specified), and *eifGetIclLibList* (returns a list of all the ICLs and levels that make up the specified **ICL release**).

10. Concluding remarks

The POWER6 microprocessor physical design was a tremendous feat to achieve on many fronts. The aggressive FO4 target and the power restrictions, coupled with technology and logistical challenges, drove the physical design and the many design methodology advancements described in this paper. The design methodology used for the POWER6 processor has set the stage for future approaches to microprocessor design. Advances in technology, as well as design point and logistical challenges, clearly indicate that more design process robustness, flexibility, and early learning are required to deliver competitive microprocessor designs to the marketplace in the future.

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

**Trademark, service mark, or registered trademark of Microsoft Corporation or Sony Computer Entertainment, Inc., in the United States, other countries, or both.

References

1. E. Leobandung, E. Nayakama, H. Mocuta, D. Miyamoto, K. Angyal, M. Meer, H. V. McStay, et al., "High Performance 65 nm SOI Technology with Dual Stress Liner and Low Capacitance SRAM Cell," *2005 Symposium on VLSI Technology, Digest of Technical Papers*, June 14–16, 2005, pp. 126–127.
2. J. D. Warnock, J. M. Keaty, J. Petrovick, J. G. Clabes, C. J. Kircher, B. L. Krauter, P. J. Restle, B. A. Zoric, and C. J. Anderson, "The Circuit and Physical Design of the POWER4 Microprocessor," *IBM J. Res. & Dev.* **46**, No. 1, pp. 27–51 (2002).
3. B. W. Curran, Y. H. Chan, P. T. Wu, P. J. Camporese, G. A. Northrop, R. F. Hatch, L. B. Lacey, J. P. Eckhardt, D. T. Hui, and H. H. Smith, "IBM eServer z900 High-Frequency Microprocessor Technology, Circuits, and Design Methodology," *IBM J. Res. & Dev.* **46**, No. 4/5, pp. 631–644 (2002).
4. V. Rao, J. Soreff, T. Brodnax, and R. Mains, "EinsTLT: Transistor Level Timing with EinsTimer," *Proceedings of the International Workshop on Timing Issues (TAU) in the Specification and Synthesis of Digital Systems*, March 8–9, 1999, pp. 1–6.
5. A. R. Conn, K. Scheinberg, and Ph. L. Toint, "A Derivative Free Optimization Algorithm in Practice," *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, 1998.
6. P. J. Camporese, A. Deutsch, T. G. McNamara, P. Restle, and D. Webber, "X-Y Grid Tree Tuning Method," U.S. Patent No. 6,205,571, March 20, 2001.
7. P. J. Restle, R. L. Franch, N. K. Norman, W. V. Huott, T. M. Skergan, S. C. Wilson, N. S. Schwartz, and J. G. Clabes, "Timing Uncertainty Measurements on the POWER5 Microprocessor," *2004 IEEE International Solid-States Circuits Conference, Digest of Technical Papers*, February 15–19, 2004, pp. 354–355.
8. M. S. Floyd, S. Ghiasi, T. W. Keller, K. Rajamani, F. L. Rawson, J. C. Rubio, and M. S. Ware, "System Power Management Support in the IBM POWER6 Microprocessor," *IBM J. Res. & Dev.* **51**, No. 6, pp. 733–746 (2007, this issue).
9. S. R. Nassif and J. N. Kozhaya, "Fast Power Grid Simulation," *Proceedings of the 37th Design Automation Conference*, 2000, pp. 156–161.
10. J. S. Neely, H. H. Chen, S. G. Walker, J. Venuto, and T. J. Bucelot, "CPAM: A Common Power Analysis Methodology for High-Performance VLSI Design," *IEEE Conference on Electrical Performance of Electronic Packaging*, Scottsdale, AZ, October 23–25, 2000, pp. 303–306.
11. R. M. Averill III, K. G. Barkley, M. A. Bowen, P. J. Camporese, A. H. Dansky, R. F. Hatch, D. E. Hoffman, et al., "Chip Integration Methodology for the IBM S/390 G5 and G6 Custom Microprocessors," *IBM J. Res. & Dev.* **43**, No. 5/6, pp. 681–706 (1999).
12. H. Smith, A. Deutsch, S. Mehrotra, D. Widiger, M. Bowen, A. Dansky, G. Kopcsay, and B. Krauter, "*R*(*f*)*L*(*f*)*C* Coupled Noise Evaluation of an S/390 Microprocessor Chip," *IEEE Conference on Custom Integrated Circuits*, San Diego, May 2001, pp. 237–240.
13. A. Deutsch, H. H. Smith, B. J. Rubin, B. L. Krauter, and G. V. Kopcsay, "New Methodology for Combined Simulation of Delta-*I* Noise Interaction with Interconnect Noise for Wide, On-Chip Data-Buses Using Lossy Transmission-Line Power Blocks," *IEEE Transactions on Advanced Packaging*, Vol. 29, February 2006, pp. 11–20.
14. K. L. Shepard and V. Narayanan, "Noise in Deep Submicron Digital Design," *1996 International Conference on Computer-Aided Design (ICCAD 1996), Digest of Technical Papers*, 1996, pp. 524–531.
15. J. R. Black, "Electromigration—A Brief Survey and Some Recent Results," *IEEE Transactions on Electron Devices*, Vol. 16, 1969, pp. 338–347.
16. J. Friedrich, B. McCredie, N. James, B. Huott, B. Curran, E. Fluhr, G. Mittal, et al., "Design of the POWER6 Microprocessor," *Proceedings of the International Solid-State Circuits Conference (ISSCC), Digest of Technical Papers*, San Francisco, CA, February 11–15, 2007, pp. 96–97.

Received February 1, 2007; accepted for publication September 18, 2007; Internet publication November 1, 2007

Rex Berridge *IBM Systems and Technology Group, 11500 Burnet Road, Austin, Texas 77850 (rexb@us.ibm.com)*. Mr. Berridge is a Senior Engineering Manager in the integration and methodology department. He received a B.S. degree in electrical engineering from Texas A&M University in 1999. He subsequently joined IBM, where he has worked on transistor-level timing. In 2005 he received an IBM Outstanding Innovation Award for his work on POWER5 transistor-level timing.

Robert M. Averill III *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (averillr@us.ibm.com)*. Mr. Averill is a Senior Technical Staff Member in the iSeries, pSeries, and zSeries* hardware development laboratory in Poughkeepsie, New York. In 1983 he joined IBM at the East Fishkill facility, where he developed advanced VLSI test equipment. He joined the advanced complementary metal-oxide semiconductor (CMOS) microprocessor group in Poughkeepsie in 1994 as a Circuit Designer and is currently the Chip Integration Leader for all iSeries, pSeries, and zSeries microprocessors. Mr. Averill received a B.S.E.E. degree from Northwestern University in 1983 and an M.S.E.E. degree from Syracuse University in 1990. He has received three IBM Outstanding Technical Achievement Awards, one IBM Outstanding Innovation Award, and one IBM Technical Corporate Award for his work in chip integration.

Arnold E. Barish *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (barish@us.ibm.com)*. Mr. Barish is a Senior Technical Staff Member working in the areas of advanced technology development, ground rules, physical verification, and library support. He received a B.S.E.E. degree from the City College of New York in 1968 and an M.S.E.E. and M.S.C.I.S. from Syracuse University in 1971 and 1977, respectively. He joined IBM in 1968 working on circuit design and I/O wiring rules and later on technology development, ground rules, physical verification, and library applications. Mr. Barish holds several patents and has received a division award for his work on H2 library development, as well as three Outstanding Technical Achievement Awards for his work on H5, POWER4, and POWER5 processor library development.

Michael A. Bowen *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (MichaelBowen@us.ibm.com)*. Mr. Bowen is a Senior Programmer in the iSeries, pSeries, and zSeries hardware development laboratory in Poughkeepsie, New York. In 1989, he joined IBM at the Kingston facility, where he worked with a team developing timing-driven placement and wiring methodologies. He joined the microprocessor team in Austin, Texas, in 1994 and continued to develop integration tools and methodologies to support the IBM RS/6000* and chips developed in collaboration with Motorola. He joined the advanced CMOS microprocessor group in Poughkeepsie in 1997 as a tool developer and is currently the Tools/Methodology Leader for zSeries systems. Mr. Bowen received a B.A. in math and computer science from the State University of New York at Potsdam in 1988 and an M.S. in computer science from Rensselaer Polytechnic Institute in 1992. He has received two Outstanding Technical Achievement and two Outstanding Contribution Awards for his work in chip integration. He also has four patents in various physical design processes.

Peter J. Camporese *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (pcamp@us.ibm.com)*. Mr. Camporese is a Senior Technical Staff Member at the IBM development laboratory, working on microprocessor physical architecture and integration. Mr. Camporese received a B.S. degree in electrical engineering from the Polytechnic University in 1988 and an M.S. degree in computer engineering from Syracuse University. He joined the IBM data systems division in Poughkeepsie, New York, in 1988, where he has worked on system performance, circuit design, physical design, and chip integration. He was the Technical Team Leader and Chief Physical Design Architect for the G4 and G7 CMOS zSeries microprocessors. He holds 12 U.S. patents and is a coauthor of several papers on high-speed microprocessor design. He has received an IBM Corporate Award for IBM eServer z900 microprocessor development and several IBM Outstanding Technical Achievement and Outstanding Innovation Awards for microprocessor physical design, integration, and tools development. He currently manages the physical design and integration development efforts for future IBM eServer microprocessors.

Jack DiLullo *IBM Systems and Technology Group, 11400 Burnett Road, Austin, Texas 78758 (dilullo@us.ibm.com)*. Mr. DiLullo is a Senior Engineer working on the integration and timing team in Austin, Texas. He received a B.S. degree in electrical engineering at Polytechnic Institute of New York in 1983 and joined IBM Austin. There, Mr. DiLullo worked in the design verification group involved in timing verification and signoff for mainframe designs. He received an M.S. degree in computer engineering from Syracuse University in 1988 and later joined IBM Boca Raton in 1994 working on IBM OS/2* operating system performance. After moving to Austin in 1996, Mr. DiLullo joined IBM EDA as an application engineer in support of IBM CMOS microprocessor designs before becoming a member of the POWER4 team working on timing methodology development and timing closure. Currently, Mr. DiLullo specializes in global timing methodology for the pSeries, zSeries, and gaming chips, as well as pSeries global chip timing closure.

Peter E. Dudley *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (pdudley@us.ibm.com)*. Mr. Dudley is an Advisory Engineer in the integration and methodology department. He received a B.S. degree in computer science and an M.S. degree in electrical engineering in 1991 and 1995, respectively, from the University of Vermont in Burlington. In 1994, he joined IBM at its Burlington, Vermont, facility and worked in the PowerPC* processor hardware development laboratory developing tools and methodologies concentrating on the data management and auditing of advanced microprocessor designs. In 1996, he joined the POWER4 processor hardware development laboratory in Fishkill and then relocated to the IBM site in Poughkeepsie. He received Outstanding Technical Achievement Awards for his audit methodology work and for his work on the POWER5 processor tools and methodology. He is the primary owner of a patent on circuit delay abstraction and is author or coauthor of three technical papers. He is currently Infrastructure Tools Leader of all IBM server and games microprocessor development projects.

Joachim Keinert *IBM Systems and Technology Group, Boeblingen Development Laboratory, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (keinert@de.ibm.com)*. Mr. Keinert received an M.S. degree in electrical engineering from the Technical University of Stuttgart, Germany, in 1980. He joined IBM in 1979 to work on bipolar circuit design. In 1982, he started to work on CMOS circuit tool development and chip design methodologies. Since then, he has been involved in the development of design tools for all IBM CMOS mainframe processors. His work also covers innovative technologies such as FinFETs and he holds several patents in various areas. Currently, Mr. Keinert is a focal point for circuit design tools for future IBM eServer processors.

David W. Lewis *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (dlewis@us.ibm.com).* Mr. Lewis is a Senior Engineer in the iSeries, pSeries, and zSeries hardware development laboratory in Poughkeepsie, New York. Mr. Lewis received a B.S. degree in computer systems engineering and an M.S. degree in computer science from Rensselaer Polytechnic Institute in 1995, and 2001, respectively. In 1995, he joined IBM at its Burlington, Vermont, facility and worked in the PowerPC processor hardware development laboratory developing circuit design tools for advanced microprocessor design. In 1996, he joined the POWER4 processor hardware development laboratory in Fishkill, New York, where he continued his work in the area of circuit design tool development. Currently, Mr. Lewis is the zSeries tools leader, along with being the physical design automation leader for all iSeries, pSeries, and zSeries microprocessors.

Robert D. Morel *IBM Systems and Technology Group, 11400 Burnett Road, Austin, Texas 78758 (rmorel@us.ibm.com).* Mr. Morel is a Senior Engineer in the iSeries, pSeries, and zSeries hardware development laboratory in Austin, Texas. In 1993 he joined IBM at its Burlington, Vermont, facility and worked in the PowerPC hardware development laboratory developing tools and methodologies for advanced microprocessor design. In 1996 he joined the POWER4 hardware development laboratory in Fishkill, New York, where he continued his work in the area of tools and methodology development. Mr. Morel received B.S.E.E. and M.S.E.E. degrees in 1992 and 1996, respectively, from the University of Vermont in Burlington. He has received an Outstanding Technical Achievement Award for his work in tools and methodology development.

Thomas Rosser *IBM Systems and Technology Group, 11400 Burnett Road, Austin, Texas 78758 (rosser@us.ibm.com).* Mr. Rosser is a Senior Technical Staff Member. He received his B.S. degree in electrical engineering at the University of Missouri at Columbia in 1976, and he joined IBM in Fishkill, New York. In his 30 years in design automation tools at IBM, he has worked in test generation, fault simulation, software simulation, hardware simulation, integrated tools development, design methodologies, circuit characterization and rules, static timing, designer productivity tools and interfaces, language parsing, design verification, logic synthesis, and physical design optimization. He holds 12 patents and serves on the Patent Review Board for the Systems and Technology Group in Austin, Texas. He currently leads the RLM flow for all IBM processors.

Nicole S. Schwartz *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (nschwart@us.ibm.com).* Miss Schwartz is a Staff Engineer in the zSeries/pSeries integration and tools department in Austin, Texas. She joined IBM in 2001 as a member of the chip integration team for the POWER family of processors. She has continued to work in unit and chip integration on the pSeries and zSeries chips with a primary focus on global clock distribution tools and methodology. Miss Schwartz received a B.S.E. in electrical engineering and computer science from Duke University in 2001 and an M.S.E. in computer engineering from the University of Texas at Austin in 2006.

Philip Shephard *IBM Systems and Technology Group, 11500 Burnet Road, Austin, Texas 77850 (shephard@us.ibm.com).* Mr. Shephard is a Senior Engineer in the PCORE/SRAM design department. He received a B.S.E.E. degree from DeVry Institute of Technology in 1977 and an M.S. degree in computer science from Union College in 1984. He joined IBM in 1978. He worked on various aspects of design for testability (DFT) through 2001 when he took an assignment to drive the implementation and bring-up of transistor-level timing analysis on SRAMs. He holds ten patents in the fields of DFT and timing analysis, with two more pending, and he has received two IBM Outstanding Technical Achievement Awards.

Howard H. Smith *2455 South Road, Poughkeepsie, New York 12601 (smithh@us.ibm.com).* Mr. Smith received a B.S. and an M.S. degree in electrical engineering from the New Jersey Institute of Technology, Newark, New Jersey, in 1984 and 1985, respectively. He joined IBM in 1984 as an integrated circuit engineer at its semiconductor development laboratory in Fishkill, New York, working in the area of high-performance gate array designs. Mr. Smith is currently a Senior Engineer at the IBM Systems and Technology group in Poughkeepsie, New York, where he is responsible for electrical analysis issues associated with high-density CMOS circuit technology and package-related products. His recent assignments include the development of on-chip noise and power grid verification processes for the IBM processor designs. His expertise lies in the area of electrical noise modeling and prediction at system-level computer operation. He has coauthored several papers on system-level noise prediction, on-chip interconnects, and electromagnetic characterization of connectors and antennas. He has several patents in his field of expertise.

Dave Thomas *IBM Systems and Technology Group, 3039 Cornwallis Road, Research Triangle Park, North Carolina 27709 (thomasdr@us.ibm.com).* Mr. Thomas is a Senior Engineer. He received his B.S. degree in electrical engineering at the University of Missouri at Columbia in 1977 and completed graduate coursework at the University of Kentucky. He joined IBM in 1977 and worked as a DRAM Circuit Designer. Over his 29-year career with IBM, he has performed in many roles including management, analog circuit design, modem design, logic design, dc/dc converter design, and tools software development. He received an Outstanding Technical Achievement Award for Smart Power development and holds seven patents in power control systems, unique circuit topologies for integrating dc/dc regulators on VLSI chips, and nonvolatile memory cells. He is currently responsible for development of power/performance/yield estimation tools for zSeries and pSeries processors.

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 Ph.D. in physics from the University of Illinois in 1986. At IBM Research, he has worked on CMOS modeling, package test, DRAM variable retention time, and high-speed interconnect modeling. For the past decade, he has concentrated on methodology, tools, and designs for high-performance clock distribution networks. He has contributed to more than a dozen high-performance microprocessors including all recent IBM mainframes, the

POWER4, POWER5, and POWER6 microprocessors, and the Microsoft Xbox 360** entertainment system and the Sony Cell Broadband Engine** processors.

John R. Ripley *IBM Systems and Technology Group, 11500 Burnet Road, Austin, Texas 77850 (rip@us.ibm.com)*. Mr. Ripley is a Senior Technical Staff Member in the iSeries, pSeries, and zSeries hardware development laboratory in Austin, Texas. He received a B.S. degree in electrical engineering from the University of Tennessee and an M.S.E.E. degree from the University of Texas in 1985. He joined IBM in 1980 and has worked on advanced CMOS microprocessor development spanning the POWER microprocessor to the current POWER6 microprocessor. Over his career with IBM, he has performed many roles including management, logic design, circuit design, DFT, integration tools and methodology development, and chip integration. He is currently the lead chip integrator for the POWER6 chip.

Stephen L. Runyon *IBM Systems and Technology Group, 11500 Burnet Road, Austin, Texas 78758 (steve@us.ibm.com)*. Mr. Runyon is a Senior Technical Staff Member working in the areas of process technology, physical design and circuit layout, yield, design for manufacturability and physical verification. He received a B.E.E. degree from the Georgia Institute of Technology in 1980 and joined IBM in 1981, where he has worked in circuit design, layout, and checking, and later in chip integration and technology. He received an M.S.E.E. degree from the University of Texas in 1985 and holds numerous patents in various areas. He has received two Outstanding Technical Achievement Awards for his work on POWER4 and POWER5 processor designs.

Patrick M. Williams *IBM Systems and Technology Group, 2070 Route 52, Hopewell Junction, New York 12533 (patricw@us.ibm.com)*. Mr. Williams is Senior Engineering Manager of the transistor-level automation department in the engineering design automation group. In 1984, he joined IBM at the East Fishkill facility, where he developed VLSI high-speed memory test systems. In 1992, he joined the advanced CMOS microprocessor team in Poughkeepsie, New York. He was initially part of the processor SRAM development team and in 1994 joined the CAD development team in support of the zSeries line of processors. He was the Lead Circuit Methodologist in support of the POWER6 processor development. Mr. Williams has been involved in many aspects of CAD development related to high-speed microprocessors, including timing, noise, power, internal resistance drop and electromigration analysis, device and parasitic extraction, chip integration, circuit optimization, electrical checking, and layout automation. He received a B.S.E.E. degree from Pennsylvania State University in 1984. Mr. Williams holds several U.S. patents and has received four IBM Outstanding Technical Achievement Awards.