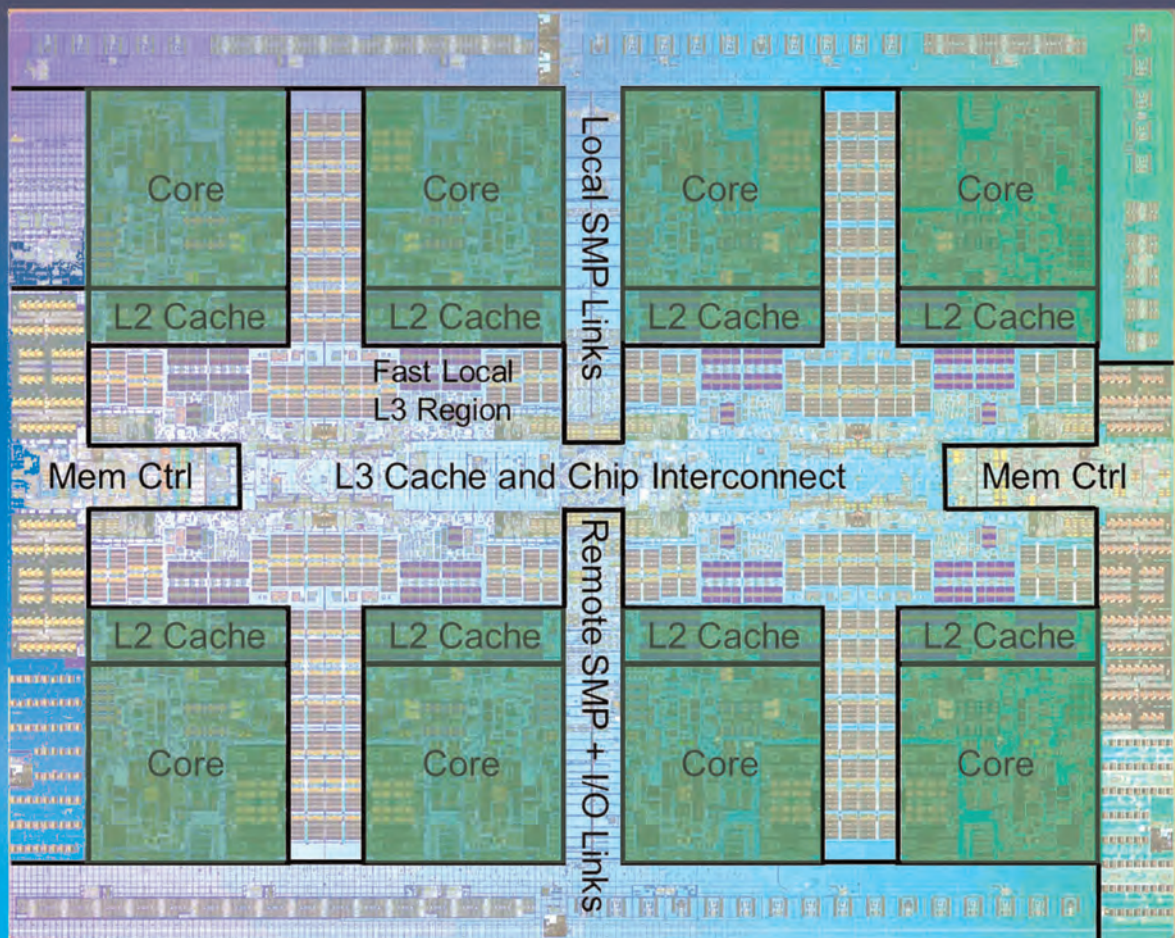


VOLUME 55, NUMBER 3, MAY/JUNE 2011

IBM

Journal of Research and Development

Including IBM Systems Journal



IBM POWER7 Technology and Systems

IBM Journal of Research and Development

Vol. 55, No. 3, May/June 2011

IBM POWER7 Technology and Systems

In the seventh generation of POWER processors, IBM introduces a balanced, high-performance eight-core multi-chip design, with large on-chip embedded DRAM and 4-way simultaneous multithreading. The papers in this issue describe the key hardware and software technologies used to build POWER7 processor-based systems from chip manufacturing technologies and architectural design and verification to systems packaging, design, and performance. A variety of systems are illustrated including a high-performance, high-productivity Productive, Easy-to-use, Reliable Computing System (PERCS). This issue includes a nontopical paper on tools to aid the development of natural language processing applications.

- 1 **IBM POWER7 multicore server processor**
B. Sinharoy, R. Kalla, W. J. Starke, H. Q. Le, R. Cargnoni, J. A. Van Norstrand, B. J. Ronchetti, J. Stuecheli, J. Leenstra, G. L. Guthrie, D. Q. Nguyen, B. Blaner, C. F. Marino, E. Retter, and P. Williams
- 2 **IBM POWER7 systems**
R. X. Arroyo, R. J. Harrington, S. P. Hartman, and T. Nguyen
- 3 **PERCS: The IBM POWER7-IH high-performance computing system**
R. Rajamony, L. B. Arimilli, and K. Gildea
- 4 **IBM POWER7 performance modeling, verification, and evaluation**
M. Srinivas, B. Sinharoy, R. J. Eickemeyer, R. Raghavan, S. Kunkel, T. Chen, W. Maron, D. Flemming, A. Blanchard, P. Seshadri, J. W. Kellington, A. Mericas, A. E. Petruski, V. R. Indukuru, and S. Reyes
- 5 **45-nm silicon-on-insulator CMOS technology integrating embedded DRAM for high-performance server and ASIC applications**
S. S. Iyer, G. Freeman, C. Brodsky, A. I. Chou, D. Corliss, S. H. Jain, N. Lustig, V. McGahay, S. Narasimha, J. Norum, K. A. Nummy, P. Parries, S. Sankaran, C. D. Sheraw, P. R. Varanasi, G. Wang, M. E. Weybright, X. Yu, E. Crabbe, and P. Agnello

(continued on next page)

© Copyright 2011 by International Business Machines Corporation. See individual articles for copying information. Post-1994 articles that carry a code at the bottom of the first page may be copied, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, U.S.A. Pages containing the table of contents may be freely copied and distributed in any form. ISSN 0018-8646. Printed in U.S.A.

(continued from previous page)

- 6 **IBM POWER7 processor circuit design**
D. F. Wendel, J. Barth, D. M. Dreps, S. Islam, J. Pille, and J. A. Tierno

- 7 **Power optimization methodology for the IBM POWER7 microprocessor**
V. Zyuban, J. Friedrich, C. J. Gonzalez, R. Rao, M. D. Brown, M. M. Ziegler, H. Jacobson,
S. Islam, S. Chu, P. Kartschoke, G. Fiorenza, M. Boersma, and J. A. Culp

- 8 **Adaptive energy-management features of the IBM POWER7 chip**
M. Floyd, M. Ware, K. Rajamani, T. Gloekler, B. Brock, P. Bose, A. Buyuktosunoglu, J. C. Rubio,
B. Schubert, B. Spruth, J. A. Tierno, and L. Pesantez

- 9 **Design methodology for the IBM POWER7 microprocessor**
J. Friedrich, R. Puri, U. Brandt, M. Buehler, J. DiLullo, J. Hopkins, M. Hossain, M. Kazda,
J. Keinert, Z. M. Kurzum, D. Lamb, A. Lee, F. Musante, J. Noack, P. J. Osler, S. Posluszny,
H. Qian, S. Ramji, V. Rao, L. N. Reddy, H. Ren, T. Rosser, B. R. Russell, C. Sze, and G. Téllez

- 10 **Functional verification of the IBM POWER7 microprocessor and POWER7 multiprocessor
systems**
K.-D. Schubert, W. Roesner, J. M. Ludden, J. Jackson, J. Buchert, V. Paruthi, M. Behm, A. Ziv,
J. Schumann, C. Meissner, J. Koesters, J. Hsu, and B. Brock

- Nontopical Paper**
- 11 **U-Compare: A modular NLP workflow construction and evaluation system**
Y. Kano, M. Miwa, K. B. Cohen, L. E. Hunter, S. Ananiadou, and J. Tsujii

IBM POWER7 multicore server processor

The IBM POWER® processor is the dominant reduced instruction set computing microprocessor in the world today, with a rich history of implementation and innovation over the last 20 years. In this paper, we describe the key features of the POWER7® processor chip. On the chip is an eight-core processor, with each core capable of four-way simultaneous multithreaded operation. Fabricated in IBM's 45-nm silicon-on-insulator (SOI) technology with 11 levels of metal, the chip contains more than one billion transistors. The processor core and caches are significantly enhanced to boost the performance of both single-threaded response-time-oriented, as well as multithreaded, throughput-oriented applications. The memory subsystem contains three levels of on-chip cache, with SOI embedded dynamic random access memory (DRAM) devices used as the last level of cache. A new memory interface using buffered double-data-rate-three DRAM and improvements in reliability, availability, and serviceability are discussed.

B. Sinharoy
R. Kalla
W. J. Starke
H. Q. Le
R. Cargnoni
J. A. Van Norstrand
B. J. Ronchetti
J. Stuecheli
J. Leenstra
G. L. Guthrie
D. Q. Nguyen
B. Blaner
C. F. Marino
E. Retter
P. Williams

Introduction

Over the years, IBM POWER* processors [1–4] have introduced reduced instruction set computing architecture, advanced branch prediction, out-of-order execution, data prefetching, multithreading, dual-core chip, core accelerators [5], on-chip high-bandwidth memory controller, and highly scalable symmetric multiprocessing (SMP) interconnect. In this seventh-generation POWER processor [6–9], IBM introduces a balanced, eight-core multichip design, with large on-chip embedded dynamic random access memory (eDRAM) caches, and high-performance four-way multithreaded cores, implementing IBM PowerPC* architecture version 2.06 [10].

Starting in 2001 with POWER4* in 180-nm technology up to POWER6* in 65-nm, spanning four technology generations, all POWER processors were designed with two cores on the processor chip. The emphasis had been to stay with two cores per chip and significantly improve per-core performance by improving the core and cache microarchitectures, as well as exploiting the socket resources by only two cores on the chip. The socket resources that have an impact on performance include chip power, various off-chip bandwidths such as input/output (I/O), memory, and SMP bandwidths, as well as socket-level cache and

memory capacity. As we add more cores to share the existing socket resources in a given technology generation, per-core performance can be negatively affected. Unlike previous generation POWER processors, in POWER7*, four times as many cores share the socket resources. Along with significant slowdown in the performance improvements from the silicon technology, this makes it more challenging to significantly improve the per-core performance of POWER7 over POWER6.

The goal of POWER7 was to improve the socket-level, core-level, and thread-level performances significantly over POWER6 while achieving all of the following in one technology generation.

- Reduce the core area and power to such an extent that four times as many cores can be placed on the processor chip in the same power envelope as that of POWER6. To reduce power, processor frequency is reduced in POWER7, while higher performance is achieved through much more emphasis on microarchitecture improvements, such as aggressive out-of-order execution, advanced four-way simultaneous multithreading (SMT), advanced branch prediction, advanced prefetching, and cache and memory latency reduction and bandwidth improvement.
- Fit the POWER7 chip in the same socket as POWER6 and utilize the same SMP and I/O buses as in POWER6

Digital Object Identifier: 10.1147/JRD.2011.2127330

© Copyright 2011 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied by any means or distributed royalty free without further permission by computer-based and other information-service systems. Permission to republish any other portion of this paper must be obtained from the Editor.

0018-8646/11/\$5.00 © 2011 IBM

(but running at higher frequency), which allows faster time to market and more continuity in POWER system designs. The memory bus has been redesigned to run at a much higher frequency using differential signaling to support the high memory bandwidth requirements of the eight high-performance cores on the processor chip.

- Remove the external Level 3 (L3) cache chips used in previous designs in order to reduce the system cost and power and save on the significant chip I/O bandwidth (and associated processor chip area) that would otherwise be needed for connecting with external L3 cache chips.
- For significant performance improvement in high-performance computing (HPC), double the floating-point capability of each POWER7 core, compared with POWER6, yielding 8× more floating-point operations (FLOPs) per cycle per processor chip.

POWER7 is fabricated in IBM's 45-nm silicon-on-insulator technology using copper interconnects. The chip area is 567 mm² and contains 1.2-billion transistors. Since each eDRAM cell provides the function of a six-transistor (6T) static random access memory (SRAM) cell, POWER7 has the equivalent function of a 2.7-billion-transistor chip.

eDRAM [11, 12] is a key technology innovation on the POWER7 chip. This allows the large 32-MB-shared-L3 cache to be on-chip, which provides several advantages such as improved L3 latency by elimination of off-chip drivers and receivers and wire-length reduction, improved L3 bandwidth per core (on-chip interconnects provide each core with 32-byte buses to and from the L3), and less area and power, compared with SRAM. In summary, on-chip eDRAM in POWER7 provides one-sixth the latency, twice the bandwidth, compared with off-chip eDRAM, and one-fifth standby power in one-third the area, compared with SRAM.

Figure 1 shows the POWER7 chip, which has eight processor cores, each with 12 execution units capable of simultaneously running four threads. To feed these eight high-performance cores, POWER7 has two memory controllers—one on each side of the chip. Each memory controller supports four channels of double-data-rate-three (DDR3) memory. These eight channels together provide 100 GB/s of sustained memory bandwidth. At the top and the bottom of the chip are the SMP links, providing 360 GB/s of coherency bandwidth that allows POWER7 to efficiently scale to 32 sockets.

In addition to high-performance cores, POWER7 contains a balanced design, with significant improvements in key server attributes, such as single-thread (ST) performance, system throughput performance, system scalability for a wide spectrum of customer workloads, energy efficiency, and excellent reliability, availability, and serviceability (RAS).

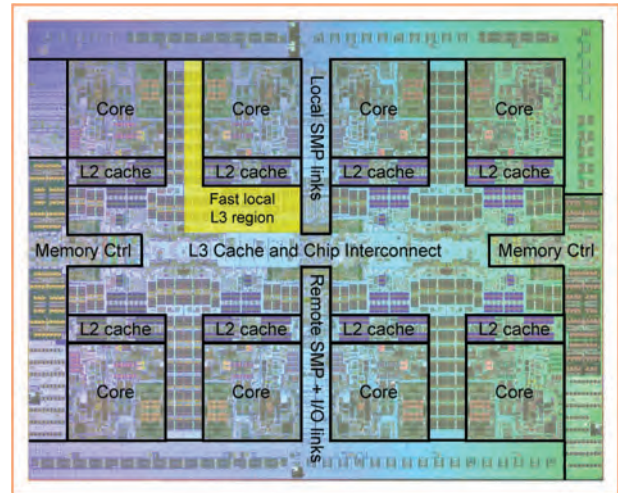


Figure 1

Die photo of the IBM POWER7 chip.

Single-thread performance is improved with deeper out-of-order execution, better branch prediction, and reduced latencies to various resources such as the caches and translation look-aside buffers (TLBs). In an ST mode, practically all the core resources can be used by the single thread.

Core throughput performance is improved by adding four-way SMT (SMT4), better sharing of core resources among the threads, and larger resource sizes. This enables POWER7 to provide increased core performance, compared with POWER6 with a smaller and more energy-efficient core.

The eight cores on the processor chip can provide 32 concurrently executing threads (8× more than POWER6), yielding very high throughput per socket. Alternatively, some cores can be turned off, reallocating energy and cache to the remaining cores for further improvement in core-level performance. Each core has its own digital phase-locked loop, allowing independent frequency control per core to match workload demands. Furthermore, turning off some threads within a core allows the core execution resources to be distributed among the remaining threads, resulting in better thread-level performance. By turning cores on and off, turning threads on and off, dynamically varying each core's operating frequency, and dynamically partitioning and reallocating the massive eDRAM cache, POWER7 is capable of meeting a wide range of rapidly changing operating requirements in an energy-efficient manner.

At the socket and system levels, advances in cache hierarchy, memory bandwidth, interconnect technology, coherence protocol, and significant core and cache power and area reduction allowed the transition from the two-core POWER6 chip to the eight-core POWER7 chip while maintaining the balance of memory and snoop bandwidth

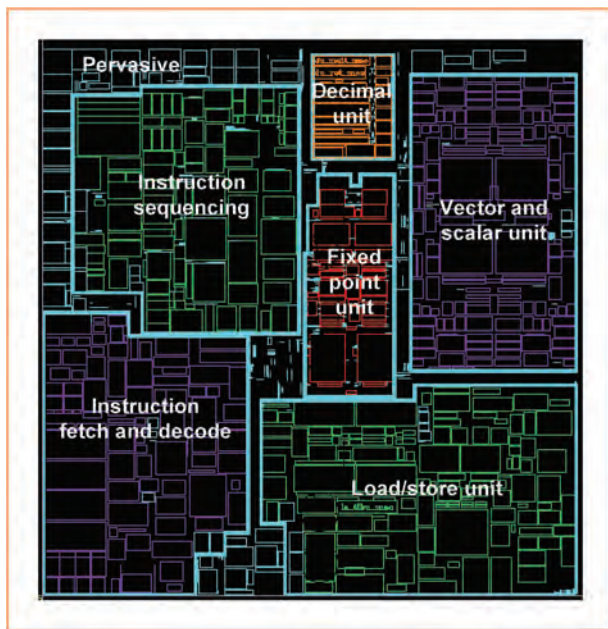


Figure 2

IBM POWER7 processor core floorplan.

per core. POWER processors and systems are well known for their balance of compute power with bandwidth as these are crucial in building large systems with high scalability.

The POWER7 design and packaging options allow POWER7 chips to be used from single-socket blades to high-end 32 socket servers, to multipetaflops clustered systems. Currently, the following three packages are being offered for POWER7 chip:

- A low-end organic package with reduced pin count bringing out one memory controller and three 5-byte SMP links to build systems with up to four sockets.
- A high-end ceramic package with two memory controllers and five 10-byte SMP links to build systems with up to 32 sockets.
- A compute-intensive package offering 32 cores per socket with double-width internal buses for HPC applications.

POWER7 core

Figure 2 shows the processor core floorplan. POWER7 has advanced branch prediction and prefetching capabilities, as well as deep out-of-order execution capability for significant ST performance gain. At the same time, it has sufficient resources to efficiently support four threads per core. The core can dynamically change mode among ST, two-way SMT (or SMT2), and SMT4 modes. **Figure 3** shows the instruction flow in the processor core.

Several POWER7 optimizations are focused on reducing core power and area. To reduce power and area, a partitioned approach to the SMT4 design was incorporated. With this approach, a pair of threads is supported from one physical general-purpose register (GPR) file that feeds one fixed-point unit (FXU) pipeline and one load/store unit (LSU) pipeline, and another pair of threads is supported from a separate physical GPR file that feeds a separate FXU pipeline and LSU pipeline. With this approach, POWER7 can efficiently rename registers for twice as many threads with a total of physical GPR file entries that is less than that of POWER5*, which only supported SMT2.

In earlier out-of-order machines, such as POWER4 and POWER5, the register rename structure for the GPR, floating-point register (FPR), and vector register (VR) was separate, which required a large number of entries. In POWER7, these were all merged into one unified rename structure with a total of 80 entries, matching the maximum number of outstanding nonbranch instructions between instruction dispatch and completion. This significantly reduces the area and power of the out-of-order machine.

In addition, in earlier machines, the issue queues for floating-point instructions and fixed-point (FX) (along with load and store) instructions were separate. In POWER7, these have been combined to reduce the area and power. The new issue queue is called unified issue queue (UQ). To achieve high frequency, the large UQ is physically implemented as two 24-entry queues, i.e., UQ0 and UQ1.

The floating-point unit (FPU) and the vector media extension (VMX) unit were separate in the POWER6 design. In POWER7, these two units are merged into one unit called the *vector and scalar unit* (VSU), which also incorporates the new vector and scalar extension (VSX) architecture that allows two-way single-instruction multiple-data (SIMD) FLOPs out of a 64-entry architected register file, with 128 bits per entry. POWER7 did not increase the number of issue ports over POWER6 but still supports the new VSX instruction sets and can execute four FX operations and eight FLOPs per cycle.

Both of the Level 1 (L1) instruction and data caches are highly banked, which allows concurrent read and write accesses to the cache, whereas an individual bank can only support either two reads or one write in a given cycle. This significantly reduces the area and power for the caches, while most of the time reads and writes (as long as they go to different banks) can occur concurrently.

Several other microarchitectural improvements were also made to reduce the core power, such as fine-grained clock gating and extensive use of high-threshold voltage transistors, use of a more register-file-based design as opposed to latch-based design whenever possible, and use of pointer-based queues instead of shifting queues to reduce active power.

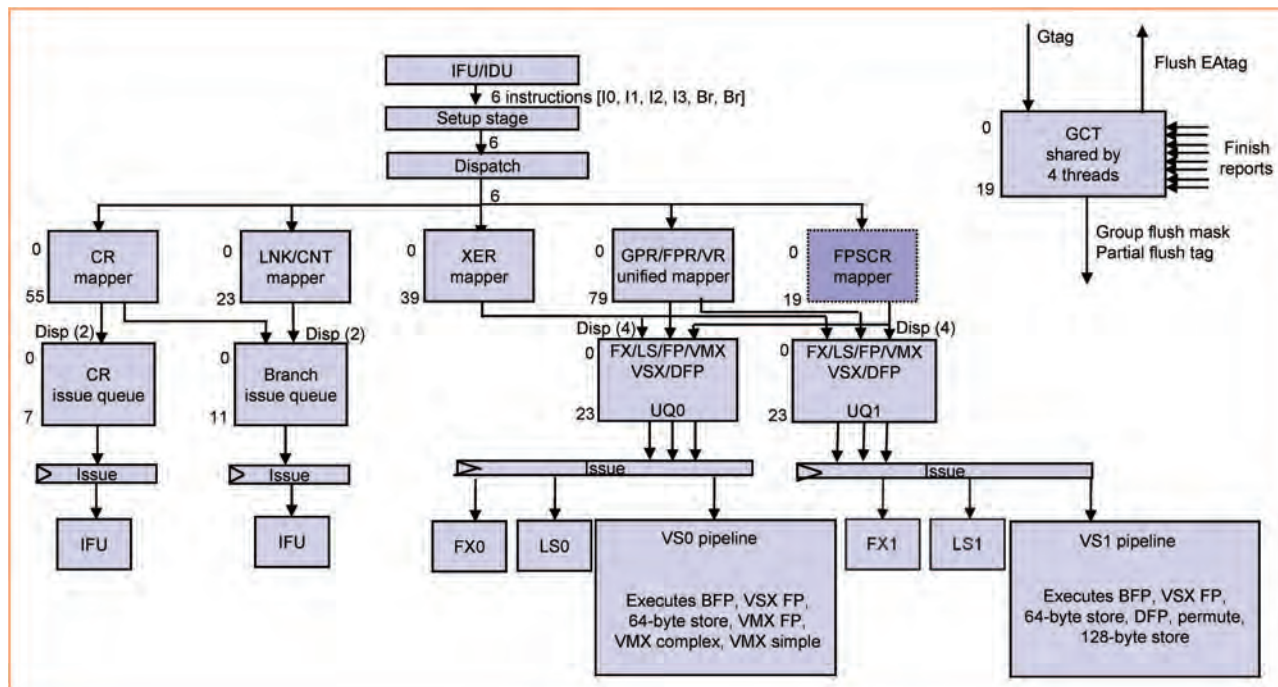


Figure 4

IBM POWER7 ISU overview.

four double-precision (DP) floating-point pipelines, one vector, one branch, one condition register (CR) logical, and one decimal floating-point (DFP) pipeline. The two LS pipes have the additional capability to execute simple FX operations. Each of the four floating-point pipelines is capable of executing DP multiply-add operations, accounting for eight FLOPs per cycle per core. The DFU, which is first introduced in POWER6, accelerates many commercial applications.

To satisfy the high bandwidth requirement of HPC workloads, the POWER7 core has twice as much LS bandwidth capability, compared with previous generations. While POWER6 can do two 8-byte loads or one 8-byte store in a given cycle, POWER7 can do two 16-byte loads and one 16-byte store in a given cycle.

For advanced virtualization capability, POWER7 has a POWER6 mode and allows dynamic partition mobility between POWER6 and POWER7 systems. Unlike previous generation processors, the TLB is not required to be invalidated in POWER7 on a partition swap. Instead, the TLB entries can be kept persistent across partition swapping so that if a partition is swapped back again, some of its translation entries may be found still in the TLB.

POWER7 allows dynamic SMT mode switches with low overhead between the ST mode (where T0 is the only thread running), the SMT2 mode (where T2 and T3

threads are not running), and the SMT4 mode of the processor core.

In addition, POWER7 implements robust RAS features. It can detect most soft errors. On soft-error detection, the core automatically flushes the instructions in the pipeline and refetches and reexecutes them so that there is no loss of data integrity. More details are provided in the section on RAS.

Instruction fetching

The IFU in POWER7 is responsible for feeding the instruction pipeline with the most likely stream of instructions, based on a highly accurate branch-prediction mechanism, from each active thread well ahead of the point of execution. The IFU is also responsible for maintaining balance of instruction execution rates from the active threads based on software-specified thread priorities, decoding and forming groups of instructions for the rest of the instruction pipeline, and executing branch instructions. **Figure 5** shows the instruction fetch and decode pipe stages in POWER7.

The POWER7 core has a dedicated 32-KB four-way set-associative I-cache. It is a 16-way banked design to avoid read and write collisions. Late select of the four ways is predicted using a 64-entry instruction effective address directory (IEADIR), which provides fast prediction for

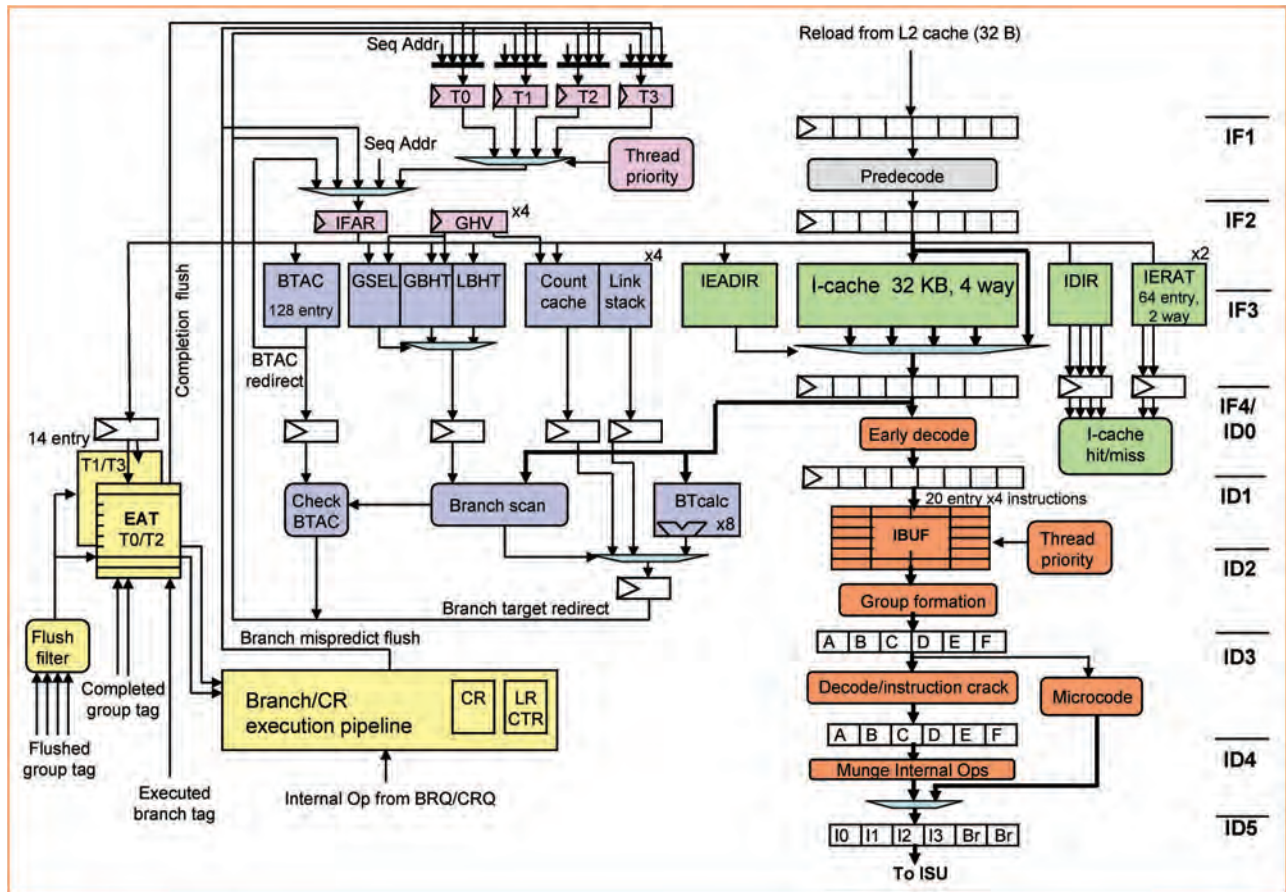


Figure 5

IBM POWER7 instruction fetch and decode pipe stages.

way selection to choose a fetch line from the four ways. A traditional full I-cache directory (IDIR) is also accessed in parallel to confirm the set selection prediction in the next cycle.

Fast address translation is supported by a 64-entry instruction effective-to-real-address translation (IERAT) table. The IERAT supports threads 0 and 2 in the first 32 entries and threads 1 and 3 in the bottom 32 entries. The IERAT directly supports 4 and 64 KB, and larger pages (64 MB and 16 GB) are supported by dynamically mapping them into 64-KB pages as needed.

The IFU fetches instructions into the L1 I-cache from the L2 unified cache. Each fetch request for instructions from the L2 returns as four sectors of 32 bytes each. These fetches are either demand fetches that result from L1 I-cache misses or instruction prefetches. For each demand fetch request, the prefetch engine initiates up to two additional L2 prefetches for the two sequential cache lines following the demand fetch. Demand and prefetch requests are made for all four instruction threads independently, and data may return

in any order, including interleaving of sectors for different cache lines. Up to four instruction fetch requests can be outstanding from the core to the L2 cache. Instruction prefetching is supported in the ST and SMT2 modes only. Up to two sequential lines are allowed to be prefetched in the ST mode and one per thread in the SMT2 mode.

When instructions are fetched from the memory subsystem, two cycles are taken to create predecode bits and parity for each of the instructions, before the instructions are written into the L1 I-cache. The predecode bits are used to scan for taken branches, help group formation, and denote several exception cases. Branch instructions are modified in these stages to help generate target addresses during the branch scan process. The modified branch instruction, with a partially computed target address, is stored in the L1 I-cache. Three cycles after the data arrives on the L2 interface, the 32 bytes are written into the I-cache. If the requesting thread is waiting for these instructions, they are bypassed around the cache to be delivered to the instruction buffers (IBUFs) and the branch scan logic.

Instruction fetch address registers (IFARs) track program counter addresses for each thread. On each cycle, the IFAR for one of the threads is selected to provide the fetch address to the I-cache complex and the branch prediction arrays. The I-cache fetch process reads up to eight instructions per cycle from the I-cache and writes them into the IBUFs where they are later formed into dispatch groups. Thread priority, cache miss pending, IBUF fullness, and thread balancing metrics are used to determine which thread is selected for fetching in a given cycle.

Branch prediction

The POWER7 core uses separate mechanisms to predict the branch direction (i.e., taken versus not-taken prediction for a conditional branch) and the branch target address. In the PowerPC architecture, the target address for a typical branch instruction can be computed from the instruction image and its address. The target address of a branch-to-link or branch-to-count instruction is architecturally available in the link or count register. Since the link or count register can be updated at any time before the execution of these instructions, the target address of these branches cannot be computed ahead of time, and hence, they need to be predicted to enable instruction fetch to stay well ahead of the point of execution.

The POWER7 IFU supports a three-cycle branch scan loop to fetch 32 bytes (or eight instructions) from the instruction cache, scan the fetched instructions for branches that have been predicted as taken, compute their target addresses (or predict the target address for a branch-to-link or branch-to-count instruction), determine whether any of these branches (in the path of execution) are unconditional or predicted as taken, and if so, make the target address of the first such branch available for next fetch for the thread.

Since it takes three cycles to obtain the next fetch address, for two of these cycles, there is no fetch for the thread. However, other active threads can utilize these cycles and do instruction fetch in the SMT modes. To reduce the loss of fetch cycles in the ST mode, POWER7 implements a branch target address cache (BTAC), described later. If the fetched instructions do not contain any branch that is unconditional or predicted taken, the next sequential address is used for the next fetch for that thread and no fetch cycles are lost.

The direction of a conditional branch is predicted using a complex of branch history tables (BHTs), consisting of an 8-K entry local BHT (LBHT) array, a 16-K entry global BHT (GBHT) array, and an 8-K entry global selection (GSEL) array. These arrays together provide branch direction predictions for all the instructions in a fetch group in each cycle. A fetch group can have up to eight instructions, all of which can be branches. These arrays are shared by all active threads. The local array is directly indexed by 10 bits from the instruction fetch address. The GBHT and GSEL

arrays are indexed by the instruction fetch address hashed with a 21-bit global history vector (GHV) folded down to 11 bits, one per thread. The value in the GSEL entry is used to choose between the LBHT and the GBHT for the direction prediction of each individual branch. All the BHT entries consist of 2 bits, with the higher order bit determining direction (taken or not taken) and the lower order bit providing hysteresis.

Branch target addresses are predicted using the following two mechanisms: 1) Indirect branches that are not subroutine returns are predicted using a 128-entry count cache, which are shared by all active threads. The count cache is indexed using an address obtained by doing an XOR of 7 bits, each from the instruction fetch address and the GHV. Each entry in the count cache contains a 62-bit predicted address along with two confidence bits. The confidence bits are used to determine when an entry is replaced if an indirect branch prediction is incorrect. 2) Subroutine returns are predicted using a link stack, one per thread. Whenever a branch-and-link instruction is scanned, the address of the next instruction is pushed down in the link stack for that thread. The link stack is “popped” whenever a branch-to-link instruction is scanned. The POWER7 link stack allows for one speculative entry to be saved in the case where a branch-and-link instruction is scanned and then flushed due to a mispredicted branch that appeared earlier in the program order. In the ST and SMT2 modes, each thread uses a 16-entry link stack. In the SMT4 mode, each thread uses an eight-entry link stack.

In the ST mode, when a taken branch is encountered, the three-cycle branch scan causes two dead cycles where no instruction fetch takes place. To mitigate the penalty incurred by taken branches, a BTAC was added to track the targets of direct branches. The BTAC uses the current fetch address to predict the fetch address two cycles in the future. When correct, the pipelined BTAC will provide a seamless stream of fetch addresses that can handle a taken branch in every cycle.

If the effect of a conditional branch is only to conditionally skip over a subsequent FX or LS instruction and the branch is highly unpredictable, POWER7 can often detect such a branch, remove it from the instruction pipeline, and conditionally execute the FX or LS instruction. The conditional branch is converted to an internal “resolve” operation, and the subsequent FX or LS instruction is made dependent on the resolve operation. When the condition is resolved, depending on the taken or not-taken determination of the condition, the FX or LS instruction is either executed or ignored. This may cause a delayed issue of the FX or LS instruction, but it prevents a potential pipeline flush due to a mispredicted branch.

Fetched instructions go to the branch scan logic and to the IBUFs. An IBUF can hold up to 20 entries, each four instructions wide. In the SMT4 mode, each thread can have

five entries, whereas, in ST and SMT2 modes, a thread can have ten entries. Special thread priority logic selects one thread per cycle for group formation. Groups are formed by reading a maximum of four nonbranches and two branches from the IBUF of the thread. Unlike the POWER4 and POWER5 processors, branches do not end groups in POWER7.

After group formation, the instructions are either decoded or routed to special microcode hardware that breaks complex instructions into a series of simple internal operations. Microcode handling continues until the architected instruction is fully emulated. The decode and dispatch section of the IFU also handles illegal special-purpose register (SPR) detection, creation of execution route bits, and marking of instructions for debugging and performance monitoring purposes.

Instruction sequencing unit

The ISU, shown in Figure 4, dispatches instructions, renames registers, issues instructions, completes instructions, and handles exception conditions.

The POWER7 processor dispatches instructions on a group basis and can dispatch a group from one thread at a time to the ISU. All resources such as the renames and various queue entries must be available for the instructions in a group before the group can be dispatched. Otherwise, the group will be held at the dispatch stage. An instruction group to be dispatched can have at most two branch instructions and four nonbranch instructions from the same thread. If there is a second branch, it will be the last instruction in the group.

Register renaming is done using the mapper logic (see Figure 4) before the instructions are placed in the issue queues. The following registers are renamed in POWER7: GPR, vector and scalar register (VSR), exception register (XER), CR, floating-point status and control register (FPSCR), link, and count. The GPR and VSR share a pool of 80 rename entries. The CRs are mapped onto 56 physical registers. The XERs are mapped onto 40 physical registers, and one nonrenamed register. The Link and Count registers are mapped onto 24 physical registers. The FPSCR is renamed using a 20-entry buffer to keep the state of the FPSCR associated with each group of instructions. Each of the aforementioned resources has a separate rename pool that can be independently accessed and shared by all active threads. Instructions that update more than one destination register are broken into subinstructions.

The ISU also assigns a load tag (LTAG) and a store tag (STAG) to manage load and store instruction flow. The LTAG corresponds to a pointer to the load-reorder-queue (LRQ) entry assigned to a load instruction. The STAG corresponds to a pointer to the store-reorder-queue (SRQ) entry assigned to a store instruction. This is also used to match the store data instruction with the store address

instruction in the SRQ. A virtual STAG/LTAG scheme is used to minimize dispatch holds due to running out of physical SRQ/LRQ entries. When a physical entry in the LRQ is freed, a virtual LTAG will be converted to become a “real” LTAG. When a physical entry in the SRQ is freed, a virtual STAG will be converted to become a “real” STAG. Virtual STAGs or LTAGs are not issued to the LSU until they are subsequently marked as being “real” in the issue queue. The ISU can assign up to 63 LTAGs and 63 STAGs to each thread.

POWER7 employs three separate issue queues: a 48-entry UQ, a 12-entry branch issue queue (BRQ), and an 8-entry CR queue (CRQ). Dispatched instructions are saved in the issue queues and then issued to the execution unit one cycle after dispatch at the earliest for the BRQ or CRQ and two cycles after dispatch at the earliest for the UQ. The BRQ and CRQ are shifting queues, where dispatched instructions are placed at the top of the queue and then trickle downward toward the bottom of the queue. To save power, the UQ is implemented as a nonshifting queue and managed by queue position pointers. The queue position pointers are shifted, but the UQ entries are not shifted, which significantly reduces the switching power in the large UQ. Instructions can issue in order or out of order from all of these queues, with higher priority given to the older ready instructions for maximizing performance. An instruction in the issue queue is selected for issuing when all source operands for that instruction are available. In addition, the STAG and the LTAG must have “real entries” for a load or store instruction before it can be issued. For the BRQ and CRQ, instruction dependences are checked by comparing the destination physical pointer of the renamed resource against all outstanding source physical pointers. For the UQ, dependences are tracked using queue pointers via a dependence matrix. The issue queues together can issue a total of eight instructions per cycle, i.e., one branch, one CR logical, two FX instructions to the FXU, two LS or two simple FX instructions to the LSU, and two vector-scalar instructions to the VSU.

The BRQ contains only branch instructions, and it receives two branches per cycle from the dispatch stage and can issue one branch instruction per cycle for execution to the IFU. The CRQ contains the CR logical instructions and moves from SPR instructions, for the IFU, the ISU, and the pervasive control unit. The CRQ can receive two instructions per cycle and can issue one instruction per cycle to the IFU.

The UQ is implemented as a 48-entry queue that is split into two halves of 24 entries each. It contains all instructions that are executed by the FXU, LSU, VSU, or DFUs. The top half of the queue contains instructions for FX0, LS0, and VS0 pipelines including VMX integer instructions. The bottom half of the queue contains instructions for FX1, LS1, and VS1 pipelines including DFP, VMX PM, and the VSU 128-bit store instructions. Appropriate instructions are

steered at the dispatch stage to the appropriate half of the UQ. The UQ can receive up to four instructions per cycle per UQ half. The 64-bit VSU store instructions are split into an address generation (AGEN) operation and a data steering operation during instruction dispatch, and a total of eight such operations can be written into UQ in a given cycle.

The relative age of the instructions in the UQ is determined by an age matrix since the UQ is a nonshifting queue, which is written at dispatch time. Each half of the UQ can issue one FX, one LS, and one VS instruction per cycle for a total of six instructions per cycle. Speculative issues can occur, for example, when an FX operation dependent on a load operation is issued before it is known that the load misses the D-cache or the data effective-to-real-address translation (D-ERAT). On a misspeculation, the instruction is rejected and reissued a few cycles later. Simple FX instructions may be selected for issue to the LSU for improved FX throughput, with the same latency as a load operation from L1 D-cache.

The ISU is responsible to track and complete instructions. POWER7 employs a global completion table (GCT) to track all in-flight instructions after dispatch. Instructions in the core are tracked as groups of instructions and, thus, will dispatch and complete as a group. The GCT has 20 entries, which are dynamically shared by all active threads. Each GCT entry corresponds to a group of instructions, with up to four nonbranch and up to two branch instructions. This allows the GCT to track a maximum of 120 in-flight instructions after dispatch. Each GCT entry contains finish bits for each instruction in the group. At dispatch, the finish bits are set to reflect the valid instructions. Instructions are issued out of order and speculatively executed. When an instruction has successfully executed (without a reject), it is marked as “finished.” When all the instructions in a group are marked “finished,” and the group is the oldest for a given thread, the group can “complete.” When a group completes, the results of all its instructions are made architecturally visible, and the resources held by its instructions are released. The POWER7 core can complete one group per thread pair (threads 0 and 2 form one pair, whereas threads 1 and 3 form the other pair) per cycle, for a maximum total of two group completions per cycle. When a group is completed, a completion group tag (GTAG) is broadcasted so that resources associated with the completing group can be released and reused by new instructions.

Flush generation for the core is handled by the ISU. There are many reasons to flush out speculative instructions from the instruction pipeline such as branch misprediction, LS out-of-order execution hazard detection, execution of a context synchronizing instruction, and exception conditions. The completion unit combines flushes for all groups to be discarded into a 20-bit mask, i.e., 1 bit for each group. The completion unit also sends out the GTAG for partial-group flushes, which occurs when the first branch is

not the last instruction in the group, and it mispredicts, causing a need to flush all subsequent instructions from the thread. A 4-bit slot mask accompanies the partial flush GTAG to point out which instructions in the group need to be partially flushed. All operations related to the canceled groups are discarded.

Data fetching

Data fetching is performed by the LSU, which contains two symmetric LS execution pipelines (LS0 and LS1), each capable to execute a load or a store operation in a cycle.

Figure 6 shows the microarchitecture for an LSU pipeline, which contains several subunits, i.e., LS AGEN and execution, SRQ and store data queue (SDQ), LRQ, load miss queue (LMQ), address translation mechanism, which includes the D-ERAT, ERAT miss queue, segment lookaside buffer (SLB) and TLB, and the L1 D-cache array with its supporting set predict and data directory (DDIR) arrays, and the data prefetch request queue (PRQ) engine.

LS execution

In the ST and SMT2 modes, a given LS instruction can execute in either pipeline. In the SMT4 mode, instructions from threads 0 and 1 execute in pipeline 0, whereas instructions from threads 2 and 3 execute in pipeline 1.

Instructions are issued to the LSU out of order, with a bias toward the oldest operations first. Stores are issued twice; an AGEN operation is issued to the LSU, whereas a data steering operation is issued to the FXU or the VSU.

Main dataflow buses into and out of the LSU include 32-byte reload data from the L2 cache and 16-byte store data to the L2 cache, 16-byte load data per execution pipeline to the VSU (with a tap off of 8-byte load data per execution pipeline to the FXU), one 16-byte store data from the VSU and 8-byte store data per execution pipeline from the FXU.

POWER7 L1 D-cache size is 32 KB, which resulted in a reduction in the D-cache access latency. FX loads have a two-cycle load-to-use latency, that is, only one cycle of “bubble” (which is a cycle in the pipeline during which no useful work is done) is introduced between a load and a dependent FXU operation. The VSU loads have a three-cycle load-to-use latency, that is, two cycles of bubbles are introduced between a load and a dependent VSU operation.

Each LSU pipeline can also execute FX add and logical instructions, allowing more FX execution capability for the POWER7 core and greater flexibility to the ISU in the issuing of instructions.

LS ordering

The LSU must ensure the effect of architectural program order of execution of the load and store instructions, although the instructions can be issued and executed out of order. To achieve that, LSU employs two main queues: the SRQ and the LRQ.

mode). During program execution, the EAs are translated by the first level translation into 46-bit real addresses that are used for all addressing in the cache and memory subsystem. The first level translation consists of two 64-entry D-ERAT cache and a 64-entry IERAT. In case of a miss in the ERAT cache (data or instruction), the second level translation is invoked to generate the translation. The second level translation consists of a 32-entry-per-thread SLB and a 512-entry TLB that is shared by all active threads.

Effective addresses are first translated into 68-bit virtual addresses using the segment table, and the 68-bit virtual addresses are then translated into 46-bit real addresses using the page frame table. While segment table and page frame tables are large and reside in main memory, a 32-entry-per-thread SLB is maintained to keep entries from the segment table to translate from effective to virtual address, and a 512-entry TLB is maintained to keep the recently used entries from the page frame table to translate from virtual to real addresses. POWER7 supports two segment sizes, i.e., 256 MB and 1 TB, and four page sizes, i.e., 4 KB, 64 KB, 16 MB, and 16 GB.

The D-ERAT is a 64-entry fully associative CAM-based cache. Physically, there are two identical copies of the D-ERAT, associated with the two LSU pipelines. In the ST and SMT2 modes, since instructions from the thread(s) can go to either LS0 or LS1 pipeline, the two copies of the D-ERAT are kept in sync with identical contents. Therefore, in the ST and SMT2 modes, logically, there are a total of 64 entries available. In the SMT2 mode, the entries are dynamically shared between the two threads. In the SMT4 mode, since the two LSU pipelines are split between the two thread pairs, the two physical copies of the D-ERAT have different contents, i.e., threads 0 and 1 dynamically share one physical 64-entry D-ERAT (associated with LS0 pipe), and threads 2 and 3 dynamically share the other physical 64-entry D-ERAT (associated with LS1 pipe), for a total of 128 logical entries. Each D-ERAT entry translates 4-KB, 64-KB, or 16-MB pages. Pages of 16 GB are installed as multiple 16-MB pages. The D-ERAT employs a binary tree least recently used (LRU) replacement policy.

The SLB is a 32-entry-per-thread fully associative CAM-based buffer. Each SLB entry can support 256 MB or 1 TB segment sizes. The multiple pages per segment (MPSS) extension of PowerPC architecture is supported in POWER7. With MPSS, a segment with a base page size of 4 KB can have 4-KB, 64-KB, and 16-MB pages to be concurrently present in the segment. For a segment base page size of 64 KB, the segment can have 64-KB and 16-MB pages concurrently. The SLB is managed by the operating system, with the processor generating a data or instruction segment interrupt when an SLB entry needed for translation is not found.

The TLB is a 512-entry four-way set-associative buffer. The TLB is managed by hardware and employs a true LRU

replacement policy. There can be up to two concurrent outstanding table-walks for TLB misses. The TLB also provides a hit-under-miss function, where the TLB can be accessed, and it returns translation information to the D-ERAT, while a table-walk is in progress. In POWER7, each TLB entry is tagged with the logical partition (LPAR) identity. For a TLB hit, the LPAR identity of the TLB entry must match the LPAR identity of the active partition running on the core. When a partition is swapped in, unlike POWER6, there is no need to explicitly invalidate the TLB entries. If the partition has previously run on the same core, there is a chance that some of its TLB entries are still available, which reduces TLB misses and improves performance.

L1 data cache organization

POWER7 contains a dedicated 32-KB eight-way set-associative banked L1 D-cache. The cache line size is 128 bytes consisting of four sectors of 32 bytes each. There is a dedicated 32-byte reload data interface from the L2 cache, which can supply 32 bytes of data in every processor cycle. The cache line is validated on a sector basis as each 32-byte sector is returned from memory subsystem. Loads can hit against a valid sector before the entire cache line is validated.

The L1 D-cache has three ports—two read ports (for two load instructions) and one write port (for a store instruction or a cache line reload). A write has higher priority over a read, and a write for a cache line reload has higher priority than a write for a completed store instruction.

The L1 D-cache consists of four physical macros organized by data bytes, each macro partitioned into 16 banks based on the EA bits, for a total of 64 banks. The cache banking allows for one write and two reads to occur in the same cycle, as long as the reads are not to the same bank(s) as the write. If a read has a bank conflict with a write, the load instruction is rejected and reissued. A 32-byte cache line reload spans eight banks, whereas a completed store instruction spans from one to four banks, depending on data length.

The L1 D-cache is a store-through design; all stores are sent to the L2 cache, and no L1 cast-outs are required. The L1 D-cache is not allocated on a store miss; the store is just sent to the L2 cache. The L1 D-cache is inclusive of the L2 cache. The L1 D-cache has byte-write capability of up to 16 bytes within a given 32-byte sector in support of store instructions.

The L1 D-cache is indexed with the EA bits. The L1 D-cache directory employs a binary tree LRU replacement policy. Being 32 KB and eight-way set-associative results in 4 KB per set, requiring up to EA bit 52 to be used to index into the L1 D-cache. A set predict array is used to reduce the L1 D-cache load hit latency. The set predict array is based on EA and is used as a minidirectory to select

which one of the eight L1 D-cache sets contains the load data. The set predict array is organized as the L1 D-cache: indexed with EA(52:56) and eight-way set-associative. Each entry contains 11 hash bits obtained from hashing bits EA(33:51), valid bits per thread, and a parity bit.

When a load executes, the generated EA(52:56) is used to index into the set predict array, and EA(33:51) is hashed and compared with the contents of the eight sets of the indexed entry. When an EA hash match occurs and the appropriate thread valid bit is active, the match signal is used as the set select for the L1 D-cache data. If there is no EA hash match, it indicates a cache miss. However, an EA hash match does not necessarily mean a cache hit. For cache hit determination, the EA is used to look up in the L1 data cache directory for the real address and then compare this real address with the real address obtained from the ERAT for the given EA.

When a cache line is validated, the default is to enter in a shared mode where all thread valid bits for the line are set. A nonshared mode is dynamically entered on an entry-by-entry basis to allow only one thread valid bit to be active. This is beneficial to avoid thrashing among the threads, allowing the same EA hash to exist for each thread at the same time.

Load miss handling

Loads that miss the L1 D-cache initiate a cache line reload request to the L2 cache, release the issue queue entry, and create an entry in the LMQ to track the loading of the cache line into the L1 D-cache and also to support the forwarding of the load data to the destination register. When the load data returns from the L2 cache, it gets higher priority in the LSU pipeline, and the data is transferred to the destination register. The LMQ is real address based and consists of eight entries, dynamically shared among the active threads. The LMQ tracks all cache line misses that result in reload data to the L1 D-cache, which also includes data prefetch requests and data touch instructions, in addition to load instructions. The LMQ supports load merging, where up to two load instructions (of the same or different threads) can be associated with a given LMQ entry and cache line reload request. The LMQ can support multiple misses (up to eight) to a given L1 D-cache congruence class.

Fixed-point unit

The FXU comprises of two identical pipelines (FX0 and FX1). As shown in **Figure 7**, each FXU pipeline consists of a multiport GPR file; an arithmetic and logic unit (ALU) to execute add, subtract, compare, and trap instructions; a rotator to execute rotate, shift, and select instructions; a count (CNT) leading zeros unit; a bit-select unit (BSU) to execute bit PM instruction; a divider (DIV); a multiplier (MULT); and a miscellaneous execution unit (MXU) to

execute population count, parity, and binary-coded decimal assist instructions. All SPRs that are local to the FXU pipeline are stored in SPR. Certain resources such as the FX XER file are shared between the two pipelines.

The most frequent FX instructions are executed in one cycle, and dependent operations may issue back to back to the same pipeline, if they are dispatched to the same UQ half (otherwise, one cycle bubble is introduced). Other instructions may take two, four, or a variable number of cycles.

At the heart of each FXU pipeline is a GPR file with 112 entries, which holds the architected registers and the renamed registers for up to two threads. The GPR has four read ports, two supplying operands for the FX pipeline, and two supplying AGEN operands to the attached LSU pipeline. Two physical write ports are clocked twice per cycle (double-pumped), giving four logical write ports, two capturing results from the two FX pipelines, and the other two from the two data cache read ports in the LSU. Double pumping the write ports reduces power consumption and the size of the GPR macro, which is important for shortening the length of critical wires that must traverse it.

Contents of the two GPR files in each pipeline are managed by the ISU to be identical in the ST and SMT2 modes but distinct in the SMT4 mode. That is, in the SMT4 mode, the GPR in one pipeline contains the architected and renamed registers for one pair of threads, whereas the GPR in the other pipeline contains the registers for the other pair of threads.

The POWER7 FXU implements a fully pipelined 64 bit $\times$ 64 bit signed/unsigned MULT with a four-cycle latency and a throughput of one 64 bit product per cycle. Since the MULT is pipelined, instructions can be issued to the FXU under a multiply operation, except when there is a result bus hazard. This level of performance is a significant improvement over preceding processors and provides considerable speedup to multiply intensive codes such as cryptographic applications. The MULT is a full custom design using Radix-4 Modified Booth Encoding and Wallace tree for partial product reduction constructed with 4 : 2 compressors. This level of MULT performance together with the extended division support added to the DIV greatly improves the performance of multiprecision arithmetic operations relative to previous POWER processors.

The latency between a compare instruction and a dependent branch instruction is often a significant performance detractor for many workloads. To reduce this latency, each FXU pipeline has a fast compare custom macro that calculates the condition code from a compare instruction faster than the ALU, resulting in a back-to-back issue in most cases for a compare, followed by a branch instruction.

To address the performance needs of several important applications, new bit-oriented FX arithmetic instructions were added to the instruction set architecture, including

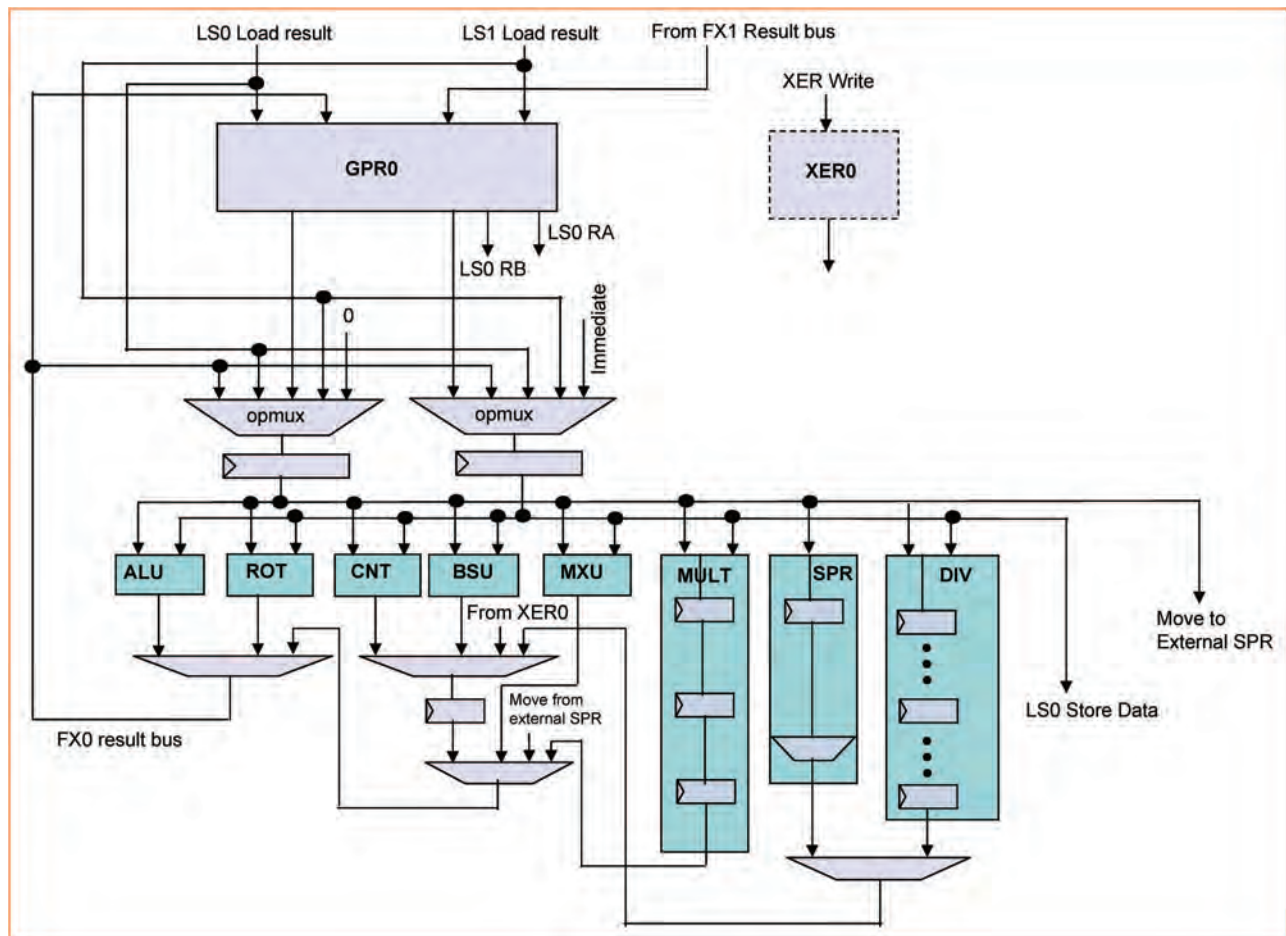


Figure 7

IBM POWER7 FXU overview (FX0 pipe shown).

64- and dual 32-bit population count and bit PM. The POWER7 FXU adds specialized execution units to process these at a throughput of one operation per cycle per pipeline with two cycle latency. A custom bit PM unit multiplexes any 8 bits from a 64-bit source register into the low-order byte of a target register. With eight such instructions and a few others to merge the result, an arbitrary permutation of a 64-bit register may be computed. In the MXU, an adder tree comprising 4 : 2 and 3 : 2 compressors computes population count on either one 64-bit, two 32-bit, or eight 8-bit quantities in two cycles, with one operation per cycle throughput, due to pipelining.

Vector and scalar instruction execution

The POWER7 VSU implements the new VSX architecture introducing 64 architected registers. With dual issue of two-way SIMD floating-point DP instructions, the performance in FLOPs per cycle per core is doubled in comparison to POWER6. In addition, the VSU of the

POWER7 processor merges the previously separate VMX unit [5] and binary FPU (BFU) [4] into a single unit for area and power reduction. Furthermore, the POWER6 DFU is attached to the VSU as a separate unit, sharing the issue port with the VS1 pipeline.

The VSU supports the following instruction types: 1) VMX/VSX PM; 2) VMX simple integer (XS); 3) VMX complex integer (XC); 4) VMX/VSX four-way vector single precision; 5) VSX two-way SIMD vector DP; and 6) scalar binary floating point. The VSU supports dual instruction issue, where VS0 pipe supports the instruction types 2–6, and VS1 pipe supports instruction types 1, 5, and 6.

The VSU implements a total of four floating-point pipelines to support the dual issue of VSX two-way SIMD vector DP floating-point instructions. Each floating-point pipeline in the VSU can execute either a 32-bit slice of types 3 and 4 or a 64-bit slice of types 5 and 6. All four floating-point pipelines are therefore needed to execute instructions of types 3 and 4 on pipe 0, but two instructions

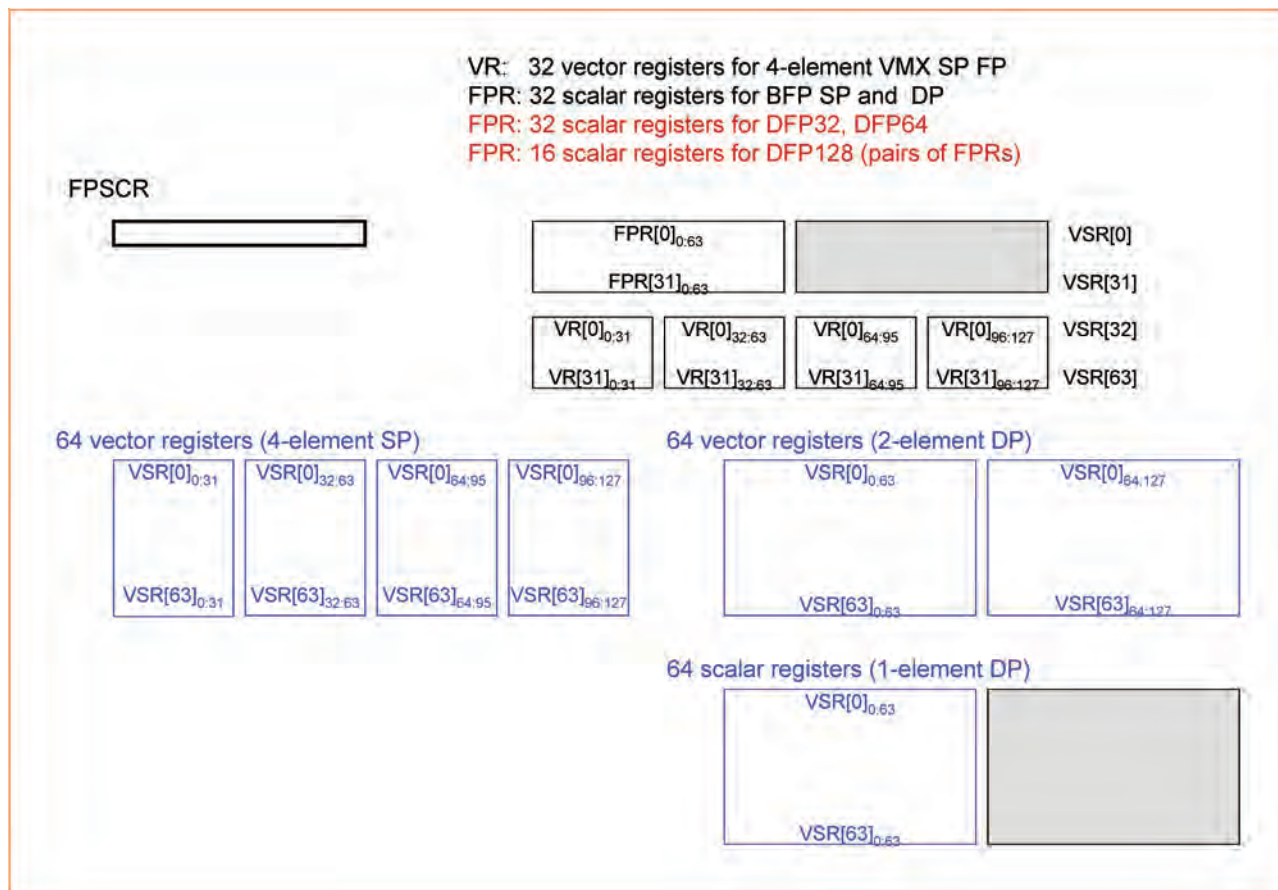


Figure 8

FPRs and VMX VRs as part of the new VSRs.

of types 5 and 6 can be simultaneously executed on both pipes using pairs of floating-point pipelines.

The new VSX instruction set architecture requires that all VSX floating point computations and other data manipulations are performed on the data residing in 64 architected 128-bit-long VSRs. This new 64-entry VSR does not increase the overall namespace of the architected registers in the core, since the existing 32 FPRs and the 32 VMX registers are mapped onto this new 64 VSRs. The mapping is shown in detail in **Figure 8**. The 32 64-bit FPRs (used for both binary and decimal FLOPs) are mapped onto bits 0 to 63 of the VSR entries 0 through 31. The 32 128-bit VRs of the VMX facility are mapped onto bits 0 to 127 of the VSR entries 32 through 63. With this mapping, the new set of VSX instructions can access all 64 registers and enable direct data exchange and data manipulations between original FPR and VMX registers.

The VSR-architected registers are physically implemented by the VSR file in the VSU. All entries in the VSR are 128 bit wide. In the physical implementation, the VSR has

172 entries, each 128 cell wide, with each cell containing 2 bits. The VSR can store 64 architected 128-bit-wide registers for all active threads (up to four), along with all of the renamed VSRs. In addition to dual bits, each VSR cell also has a multiplexer for each read port for selecting the read from the odd or even bits based on the thread identity. In the ST mode, only a single bit is in use by each cell. In the SMT2 mode, bit 0 stores thread 0 data, and bit 1 stores thread 1 data. In the SMT4 mode, threads 0 and 2 are fixed allocated to bit 0 of the dual bit cell, and threads 1 and 3 are fixed allocated to bit 1 of the dual bit cells. With 172 entries of double bit cells, in the SMT4 mode, each thread pair with its two times 64 architected registers still has 44 renames available, with a maximum of 80 renames among both thread pairs as the maximum supported by the ISU. The dual issue of instructions with three 128-bit-wide input operands for each instruction, two 128-bit-wide results buses and two 128-bit-wide load buses requires a VSR with six read and four write ports, as shown in **Figure 9**.

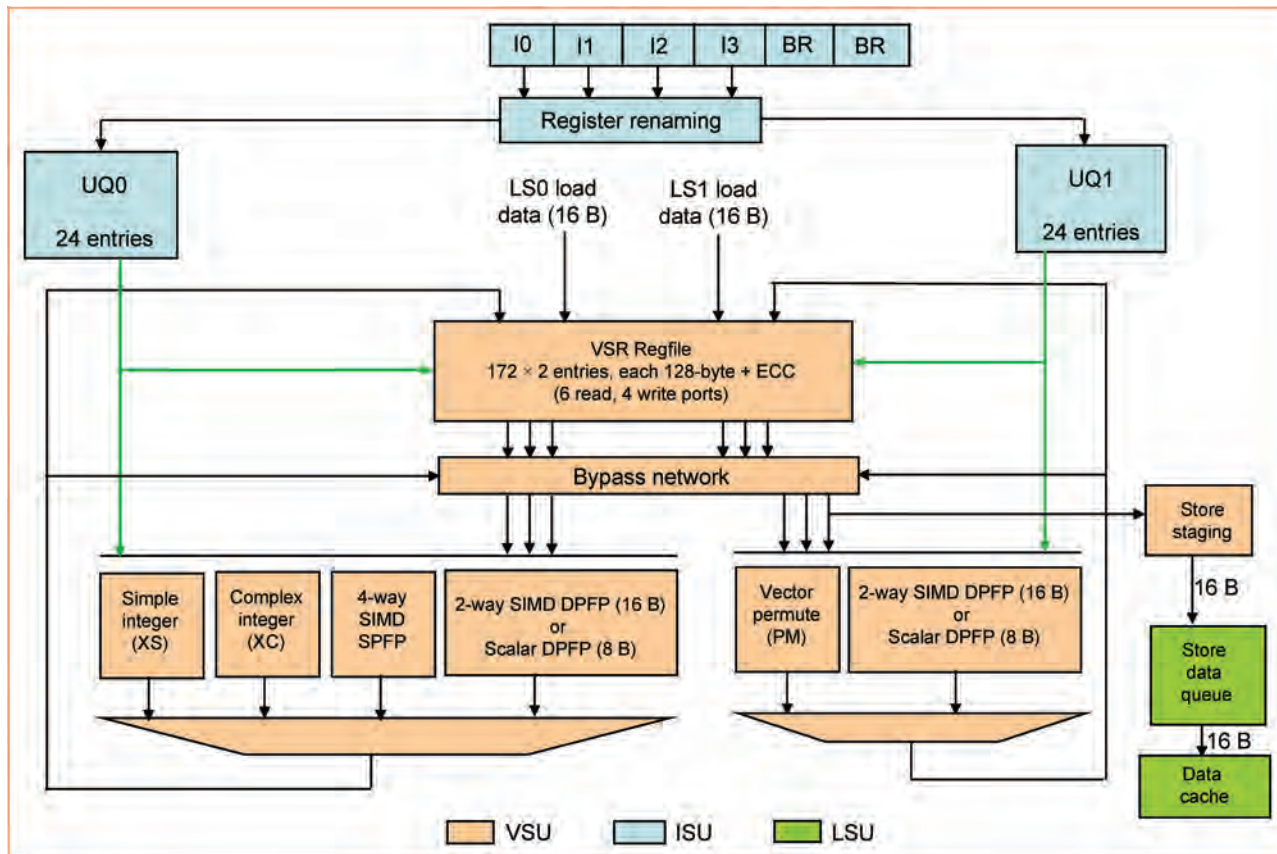


Figure 9

VSU pipeline diagram. (DPFP: double-precision floating point.)

The VSU pipeline shown in Figure 9 has the following logical execution components: 1) the XS; 2) the XC; 3) the vector PM; and 4) the floating-point pipelines that execute the scalar binary floating point, the vector single precision floating point, and the vector DP floating point. All VSU execution pipes are fully pipelined.

The XS is responsible for the vector simple FX calculations. The basic structure of XS is the same as in POWER6 [4], but the implementation has been optimized to reduce the pipeline length from three to two cycles. The vector select (VSEL) instruction is no longer part of the XS and is now part of the vector PM on pipe 1 to enable the execution of VSEL instructions in parallel with the FX instructions on pipe 0.

The PM performs permutation as well as rotates and shifts by bytes and bits on a 32-byte vector, as given by $2 \times$ a 16-byte input operand. The PM unit is the only unit requiring a local 128-bit-wide dataflow as all other units can be partitioned into $4 \times$ a 32-bit or $2 \times$ a 64-bit dataflow. A new implementation structure for PM enabled the PM overall latency to be reduced from four to three cycles in comparison with the POWER6.

To double the FLOPs per cycle in comparison with the POWER6 processor core, four fused multiply-add binary floating-point DP pipelines have been implemented in the VSU. The fast six-cycle result forwarding capability within a floating-point pipeline and between floating-point pipelines are maintained, as in previous POWER processors. The same floating-point pipelines are used to execute the newly introduced VSX vector DP floating-point instructions, the original binary scalar floating-point instructions as well as the original VMX single-precision vector floating-point (VF) and VMX complex integer (XC), for SIMD integer multiply, multiply-add, multiply-sum and sum-across. For the VF and XC implementation, all four 64-bit floating-point pipelines are used, but they operate on half the data width as needed for the four times 32-bit operation that make up the 128-bit-wide result. The issue of an instruction on VS0 that uses all four floating-point pipelines by the VF, XC, and VSX vector single precision instructions implies that no instruction can be issued at VS1 in the same cycle that uses the any floating-point pipeline. The XS and the VF have a five-cycle difference in latency but share

Table 1 Comparison of POWER6 and POWER7 cache hierarchies.

<i>POWER6 (assuming 5-GHz core)</i>	<i>POWER7 (assuming 4-GHz core)</i>
	32 KB store-through L1 D-cache 0.5ns latency, 192 GB/s private
64 KB store-through L1 D-cache 0.8ns latency, 80 GB/s private	256 KB store-in L2 cache 2.0-ns latency, 256 GB/s private
4 MB store-in L2 cache ~5.0-ns latency, 160 GB/s private	4 MB partial victim local L3 region ~6.0-ns latency, 128 GB/s private
32 MB victim L3 cache ~35-ns latency, 80 GB/s shared by 2	32 MB adaptive victim L3 cache ~30-ns latency, 512 GB/s shared by 8

the result write-back bus. The ISU prevents an XS instruction from being issued five cycles after the issue of a VF instruction on the VS0 pipeline to avoid result bus conflict.

The POWER7 VSU floating-point pipeline replaces two BFUs, VMX VF, and the VMX XC logic in the POWER6 by a single common dataflow. The execution of all these different types of instructions on a single floating-point dataflow resulted in a significant area and power reduction. In POWER7, different data formats are used by different types of instructions, increasing the complexity at the front and back ends of the floating-point pipeline. A complete redesign of the floating-point pipeline balanced the logic contents of each stage and significantly reduced the overall area and power for the floating-point pipeline, compared with previous FPU designs.

Another complexity of merging the VMX and BFU (and newly introduced VSX) is the VSU-level bypassing of the execution results with different data widths and formats to the operand latches. Despite all the complexities of the different bypass cases, the bypassing logic is designed in a way such that a given instruction has a uniform issue-to-issue latency to all other units, irrespective of the dependent instruction. The only exception is the fast forwarding path of the floating-point pipelines. It takes six cycles to forward floating-point results to any other floating-point pipeline, but it takes seven cycles if the dependent instruction is not a floating-point pipeline. The VSU instruction execution latencies are two cycles for the XS, three cycles for the PM, six cycles for floating-point fast forwarding, and seven cycles for the XC. Finally, three cycles after the issue of a load with a L1 D-cache hit, a dependent VSU instruction can be issued. With this bypassing capability and the ability to do two 128-bit loads in parallel with the dual issue of VSX instructions (such as fused multiply-add), the VSU is well balanced for HPC and for vector computing for commercial applications.

Cache hierarchy

The cache hierarchy for POWER7 has been reoptimized with regard to the prior generation POWER6 processor

in order to suit the following changes in the core and at the chip:

1. Repipelining of the core from high-frequency design to the power/performance optimized design point.
2. Change from a primarily in-order to a primarily out-of-order instruction scheduling policy.
3. Growth from two to four threads per core.
4. Reduction in L1 D-cache size from 64 to 32 KB along with reduction in L1 cache access time.
5. Growth from two cores to eight cores per die.

As shown in **Table 1**,¹ the 64-KB L1 D-cache and 4-MB L2 cache in the POWER6 processor have been replaced by a 32-KB L1 D-cache, a 256-KB L2 cache, and a 4-MB local L3 region in the POWER7 processor, essentially utilizing three layers where two had been used before [4].

This was possible because the slower POWER7 instruction pipeline and the out-of-order scheduling policy enable reduced access latencies to both the L1 D-cache and L2 cache, provided by corequisite reductions in capacity commensurate with the desired latencies.

The POWER7 cache microarchitects have exploited this opportunity to provide a substantial L2 cache latency reduction by reducing the capacity from 4 MB to 256 KB. Given the L2 cache's role in absorbing the store-through traffic from the highly threaded high-performance core, this reduction in capacity enables a corresponding reduction in the energy usage driven by the high traffic rate. Therefore, the introduction of this intermediate cache layer provides both a latency reduction and an energy reduction.

The traffic reduction provided by the 256-KB L2 cache further affords the opportunity to optimally utilize IBM's high-density low-power on-processor-chip eDRAM technology for the next level of cache. Developed for this express purpose, it has been exploited by the POWER7

¹Both POWER6 and POWER7 systems have a wide range of product offerings at different processor core frequencies. The frequencies used in the table are for the purpose of illustration.

Table 2 IBM POWER7 cache states.

<i>State</i>	<i>Description</i>	<i>Authority</i>	<i>Sharers and scope</i>	<i>Source data</i>	<i>Data cast-out</i>	<i>Scope cast-out</i>
I	Invalid	None	N/A	N/A	N/A	None
ID	Deleted, do not allocate	None	N/A	N/A	N/A	None
S	Shared	Read	Yes, scope unknown	No	No	None
SL	Shared, local data source	Read	Yes, scope unknown	At request	No	None
T	Formerly MU, now shared	Update	Yes, probably global	If notified	Yes	Required, global
TE	Formerly ME, now shared	Update	Yes, probably global	If notified	No	Required, global
M	Modified, avoid sharing	Update	No	At request	Yes	Optional, local
ME	Exclusive	Update	No	At request	No	None
MU	Modified, bias toward sharing	Update	No	At request	Yes	Optional, local
IG	Invalid, cached scope-state	None	N/A, probably global copies	N/A	N/A	Required, global
IN	Invalid, scope predictor	None	N/A, probably local copies	N/A	N/A	None
TN	Formerly MU, now shared	Update	Yes, local	If notified	Yes	Optional, local
TEN	Formerly ME, now shared	Update	Yes, local	If notified	No	None

cache microarchitects to create a low-latency 4-MB local L3 region, tightly coupled to the private L2 associated with each core. Each storage element in the local L3 region requires only one transistor, instead of the six transistors required by conventional dense SRAM technologies. The resulting area and energy reductions are a primary factor enabling the incorporation of eight cores into a single, balanced, processor chip.

The POWER6 cache hierarchy includes a shared 32-MB L3 cache, which is built from a prior eDRAM technology that requires it to be located on one or two chips that are attached to the processor chip. Given the bandwidth required to satisfy each core, using an off-chip cache shared by the eight cores on a POWER7 chip would have required a prohibitive number of off-chip signaling resources. Therefore, the POWER7 cache hierarchy includes a shared 32-MB L3 cache comprised of the 4-MB local L3 regions from the eight cores that reside on the processor chip. This enables the signaling resources that would have been used for communication with an off-chip L3 cache to be redeployed for other purposes, such as the memory and I/O subsystem interfaces.

The adaptive L3 cache management policy routes instructions and data to the 4-MB local L3 regions of the core

or cores that most actively utilize them, enabling them to experience reduced latency, even cloning copies when multiple cores are actively sharing them but also destroying copies as they fall into disuse in order to preserve capacity. Since the POWER7 32-MB L3 cache resides on the processor chip, it provides lower latency access (even to 28 MB composed of nonlocal L3 regions) than the POWER6 off-chip 32-MB L3 cache.

As shown in **Table 2**, the POWER7 L2 and L3 caches support the same 13-cache-state protocol as the POWER6 design point [4]. While no new cache states have been added for POWER7, new coherent operations are supported. One such operation is cache injection. An I/O device performing a direct memory access (DMA) write operation may target the operation to the cache, instead of to memory. If a given core's L2 cache or local L3 region owns a copy of the targeted cache line (i.e., holds the line in an M, ME, MU, T, TE, TN, or TEN cache state), the data will be installed into the local L3 region. Additionally, new heuristics have been developed, which further exploit the semantic content reflected by the existing cache states. One of these, which is called the partial victim cache management policy, reduces energy usage as data moves between a given L2 cache and its associated local L3

region. This feature is further described in the L2 cache section.

Another such enhancement involves the L3 cache management policy. While the POWER7 L2 replacement policy is similar to the POWER6 policy, employing an enhanced pseudo-LRU replacement algorithm, with biasing toward various invalid locations, the POWER7 L3 replacement policy incorporates significant enhancements and exploits the semantic content of the 13 cache states to manage the aggregation of the 4-MB regions. This feature is described in further detail in the L3 cache section.

Finally, the barrier synchronization register (BSR) facility originally implemented in POWER5 and POWER6 processors has been virtualized in the POWER7 processor [2]. Within each system, multiple megabytes of main storage may be classified as BSR storage and assigned to tasks by the virtual memory manager. The BSR facility enables low-latency synchronization for parallel tasks. Writes to BSR storage are instantaneously broadcast to all readers, allowing a designated master thread to orchestrate the activities of workers threads in a low-latency fine-grained fashion. This capability is particularly valuable for improving parallel speedups in certain HPC environments.

L2 cache

The private 256-KB POWER7 L2 cache utilizes 128 byte lines and is eight-way set associative. It is comprised of a single centralized controller, with two address-hashed cache data arrays. Some of the structures in the L2 cache operate at core frequency, while others operate at half of the core frequency. Hereafter, the terms *core cycle* and *2 : 1 cycle* will be used to distinguish between them. The interfaces into the core, the data flows, and the cache directory SRAM cells operate at the core frequency, whereas the address flows, control logic, and cache data array SRAM operate at half of the core frequency. This unique blend of core-cycle structures and 2 : 1 cycle structures optimizes coherence and data bandwidth as well as latency while reducing energy, area, and wiring congestion.

Utilizing a dual-port-read core-cycle directory SRAM enables the coherence dispatch logic to process one even cache line 128-byte snoop, one odd cache line 128-byte snoop, and one 128-byte core operation or one directory write operation every 2 : 1 cycle. Up to 64 byte may be read from or written to each of the two cache data arrays every 2 : 1 cycle. Therefore, the following 128-byte operations all utilize a given cache data array for two 2 : 1 cycles: 1) core fetches; 2) cast-out reads; 3) snoop intervention reads, and 4) write-backs of miss data, as well as read-modify-write operations (performed on behalf of accumulated core stores) that affect more than 64-byte aligned values. A single centralized scheduler manages coherence dispatches, directory writes, cache data reads, and cache data writes.

To reduce latency for sensitive core fetch operations (e.g., L1 D-cache misses and L1 instruction cache misses) and to reduce resource usage for core store operations, core fetch operations and the read portions of read-modify-write operations access the cache data arrays speculatively. However, energy usage is minimized by either canceling the second 2 : 1 cycle of a two-cycle access in the case of a cache miss or targeting the cache read to a subset of the set-associative members during the second 2 : 1 cycle of a two-cycle access in the case of a cache hit. Energy usage is further reduced in the case of read-modify-write operations as follows: Only the 8-byte double words that will be partially updated are read from the cache arrays, and only the 8-byte double words that are partially or fully updated are written to the cache arrays.

To further reduce latency, given that data reads and writes may utilize a given cache data array for two 2 : 1 cycles, the scheduler not only gives precedence to core fetch operations but also enables them to interrupt other two 2 : 1 cycle operations already in progress.

Store-through traffic from the core represents the bulk of the traffic managed by the L2 cache. A fully associative 16-deep 32-byte entry store cache absorbs every individual store executed in the core or up to 16 bytes of store traffic every core cycle. Up to four of the 32-byte store cache entries, comprising updates to the same 128-byte coherence granule, can be grouped together into a single simultaneous coherence dispatch by the L2 scheduling logic.

To further reduce energy usage, the read/claim (RC) machines, which have been utilized in prior designs to manage core fetches, read-modify-write operations on behalf of core stores, and core prefetch operations, have been divided into two classes: 1) a set of eight fully functional RC machines that are incorporated into the L2 cache controller and manage core fetches and stores that hit the L2 cache and 2) a set of 24 more energy-efficient “lightweight” RC machines that are incorporated into the L3 cache controller and manage prefetches and stores that miss the L2 cache. These lightweight RC machines stage data through the L3 cache and will be described more fully in the section on L3 cache.

This division between primarily low-latency/tenure operations in the L2 RC machines and primarily high latency, more plentiful operations in the L3 lightweight RC machines, as well as the reduction in tenure of L2 RC machine operations due to the eight core-cycle best case L2 latency and speculative cache read for both load and store read operations, has enabled the cache microarchitects to reduce the number of L2 RC machines and associated cast-out machines to eight, thereby reducing energy usage.

The L2 RC machines and L3 lightweight RC machines aggressively reorder storage operations in order to fully exploit the relaxed rules prescribed by the PowerPC storage architecture, thereby enabling the high volume of system

coherence traffic needed to scale robustly to a large number of sockets.

The symbiotic relationship between the fully functional L2 RC machines and the lightweight L3 RC machines is possible because of the tight coupling between a given 256-KB L2 cache and its corresponding 4-MB local L3 region, as well as the partial victim cache management policy that governs their interactions. At times, they behave in a traditional exclusive victim-cache manner to better manage capacity and enjoy the sum of their set associativity, and at other times, they behave in the manner of an inclusive cache hierarchy in order to reduce energy usage.

For example, when certain types of operations move a given cache line in a given subset of the cache states from the 4-MB local L3 region to the 256-KB L2, instead of invalidating the L3 copy, they leave a shared copy. Later, when the line ages out of the L2 cache, the cast-out machine first queries the L3 directory. If the residual shared copy still exists, then there is no need to read the data from the L2 cache arrays, move it to the L3 cache, and write it to the L3 cache arrays. Instead, a single write to the L3 directory updates the state of the line in the L3, thereby saving the energy that otherwise would have been expended.

L3 cache

The shared 32-MB POWER7 L3 cache is composed of eight 4-MB L3 regions. Each L3 region utilizes 128-byte lines and is eight-way set associative. A given L3 region comprises a single centralized controller, with four address hashed eDRAM cache data banks and four address hashed SRAM directory banks. The local L3 region is tightly coupled to the L2 cache associated with a given core. All L3 constructs operate in 2 : 1 cycles (defined in the L2 cache section).

A 4-MB L3 region is comprised of 32 ultradense high-speed eDRAM macros. The eDRAM macro has access latency and cycle time characteristics slightly worse than conventional 6T SRAM. As such, the combined effects of the eDRAM access latency, the overhead of waiting on the directory result before accessing the cache (to reduce energy usage), and the overhead of traversing the L2 cache prior to accessing the L3 cache are negligible. They are more than counterbalanced by the beneficial latency of the 256-KB L2 cache. Likewise, the slightly higher cycle time and reduction in overall bandwidth per unit of capacity is counterbalanced by the traffic reduction afforded by the 256-KB L2 cache. The refresh overhead, typically associated with DRAM, is hidden by a parallel engine that refreshes unused subarrays within each macro whenever operations exercise the macro.

In stark contrast to conventional SRAM, the eDRAM macro requires only one third the area and dissipates only one fifth of the standby energy of an equivalent SRAM macro. Less than 15% of the area of the POWER7 die is

consumed by the eDRAM macros comprising the 32-MB L3 cache, enabling not only eight powerful cores, but the incorporation of high-bandwidth off-chip and on-chip interconnects necessary for insuring robust system balance and scalability.

At a higher level, by incorporating the eDRAM L3 cache on the processor die, a 4-MB local L3 region enjoys roughly one sixth the access latency of an off-chip L3 cache, eliminating the need for a separate 4-MB on-chip SRAM cache to provide equivalent low latency access. Instead of incurring prohibitive off-chip bandwidth requirements to support eight cores, the on-chip L3 cache provides twice the bandwidth per core despite supporting four times as many cores.

The centralized L3 region controller provides a single core/L2 dispatch port, a single lateral L3 region dispatch port, dual snoop dispatch ports (for even and odd cache lines), and a single pool of operational resources.

Storage accesses that miss the L2 cache access the 4-MB local L3 region via the core/L2 dispatch port. Those that hit in the 4-MB local L3 region are managed by a pool of eight read machines. Prefetches and some L2 store misses also access the L3 region via the core/L2 dispatch port and are managed by a pool of 24 lightweight RC machines. When prefetched data is staged to the L3 cache, it is managed by a pool of ten write machines. Each write machine has an associated cast-out machine to manage the eviction of lines displaced by writes. Note that these write machines are also utilized by other operations such as cast-outs and cache-injections.

Storage accesses that miss both the L2 cache and the 4-MB local L3 region are broadcast to the coherence fabric and snooped by the memory controller, other L2 caches, possibly other L3 caches, and by the seven remote 4-MB L3 regions that comprise the remaining 28 MB of the on-chip L3 cache. Therefore, operations that miss the 4-MB local L3 region but hit in the remaining 28 MB of the L3 cache access the cache via the snoop dispatch ports. L2 cast-out operations access the 4-MB local L3 region via the core/L2 dispatch port, whereas lateral cast-outs access a given 4-MB L3 region via the lateral L3 region dispatch port (see targeted cases below) or via the snoop dispatch ports (see state merge cases below). All of the cast-out scenarios are described in detail in the subsequent paragraphs.

As illustrated in **Figure 10**, the capacity, data flow logic, and control flow logic of a 4-MB local L3 region are shared between the L2 cache to which the local L3 region is coupled, and the other seven 4-MB L3 regions on the chip. The associated L2 cache utilizes the 4-MB local L3 region directly as a victim cache. The other seven 4-MB L3 regions on the chip also use the 4-MB local L3 region as a victim cache under certain circumstances. Note that when the core and L2 associated with a given L3 region are disabled, the remaining L3 regions enjoy full access to the L3 region.

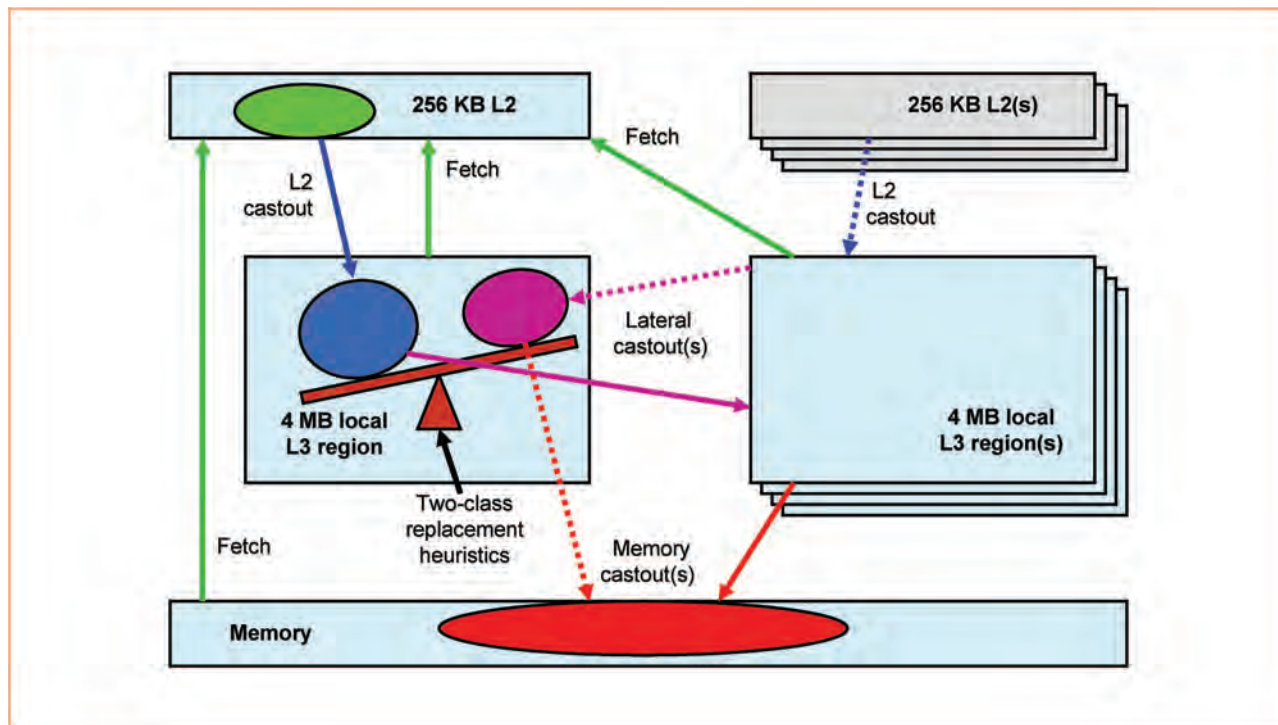


Figure 10

L3 region behavior.

When the associated core and L2 are enabled, a number of heuristics cooperate to give precedence to the associated L2 cache while allowing load and capacity balancing among all eight L3 regions on the chip, thereby causing them to behave as a highly set-associative (64-way) adaptive 32-MB L3 cache.

The replacement algorithm serves as a primary heuristic in creating this balance. It categorizes all lines in the L3 cache into the following two classes:

1. Those installed as victims via a cast-out operation by the associated L2 cache.
2. Those installed as victims via a lateral cast-out operation by one of the other seven L3 regions, those that are residual shared copies of lines that have been moved to the local L2 cache, and those that are invalid.

For a given set in the cache, a fully ordered list is maintained for each class. When a cast-out from the associated L2 needs to allocate a line, it preferentially selects the LRU line from the second class, but if no second-class lines exist, it selects the LRU line from the first class. Note that the cast-out will not need to allocate a line if it finds a copy of itself or if it finds an invalid slot. If the line finds a copy of itself (see the case described in the

L2 cache section), energy usage is reduced since no data is moved, and the cache states of the two copies are merged. If the line finds an invalid slot, it allocates without displacing another line. L2 cast-out operations are managed in the L3 region by a pool of ten L3 write machines and ten associated L3 cast-out machines.

When a lateral cast-out from an L3 region finds a copy of the line (in a subset of the valid states) in any of the other L3 regions within the lateral cast-out broadcast scope, the states of the two copies are merged at the destination, and no data is moved, saving energy (similar to the L2 cast-out case). Otherwise, a set of heuristics weigh the usage pattern (both capacity usage and traffic volume) of the targeted region to determine whether to accept the lateral cast-out. If the lateral cast-out is accepted by the targeted L3 region and needs to allocate a line (i.e., it does not find an invalid slot), it also preferentially selects the LRU line from the second class, only displacing a first-class line if no second-class lines exist. In order to allow for growth in the set of candidate second-class lines, whenever a lateral cast-out is accepted by a given L3 region, the LRU first-class line begins a transformation that will eventually convert it to a most recently used second class, thereby gradually enlarging the supply of second-class lines. Lateral cast-out operations that move data to a targeted L3 region are

managed in the L3 region by a pool of ten L3 write machines and ten associated L3 cast-out machines.

When a line is evicted from an L3 region (due to an L2 cast-out, a lateral cast-out from another L3 region, a prefetch operation, or a cache injection operation), if the evicted line is second class and has a “dirty” state, it is written back to memory. If the evicted line is first class, a set of heuristics weigh the cache state and the usage pattern (capacity) of the evicting cache to determine whether to laterally cast-out the line to another L3 region or to immediately write it back to memory (if it is “dirty”). If the line is laterally cast-out, another set of heuristics determine which of the other seven L3 regions to target based upon past history, reference pattern, and system firmware control. L3 cast-out operations are managed in the L3 region by a pool of ten L3 cast-out machines.

Memory subsystem

Each POWER7 chip includes two integrated memory controllers, each supporting up to four memory channels. A given memory channel can be connected through an off-chip interface to a memory buffer chip.

Supplying ample memory bandwidth and capacity to support a balanced eight-core chip presents a significant challenge. To meet this challenge, the POWER7 memory subsystem microarchitects employed a multifaceted strategy:

1. Exploiting the elimination of the off-chip L3 cache interface by allocating more off-chip signaling resource to the memory interface.
2. Developing a strategic high-frequency differential signaling technology to provide increased bandwidth per signal while reducing energy usage.
3. Employing a lower overhead cyclic redundancy check (CRC)-based error detection protocol.
4. Developing a hierarchical buffering scheme that provides direct access to two memory ports from each buffer chip.
5. Exploiting DDR3 memory technology to provide increased memory access frequency at the expense of more complex scheduling rules and more complex error correction codes (ECCs).
6. Developing advanced scheduling algorithms with increased reordering capability to achieve higher interface utilization despite the more complex DDR3 scheduling rules and more complex ECC.
7. Allocating more buffering resource to provide a larger pool of read and write operations to the advanced scheduler.

While the POWER6 chip allocates roughly 400 off-chip signal I/Os to the memory interface, the POWER7 chip increases this to roughly 640. The lower frequency interface utilized by the POWER6 memory subsystem has been replaced by a 6.4-GHz high-speed differential signaling

interface. Additionally, the inline ECCs utilized by the POWER6 channel have been replaced by a lower overhead CRC for the POWER7 channel. The net effect is an increase in raw signal bandwidth from 75 GB/s with the POWER6 interface to 180 GB/s with the POWER7 interface.

By providing a dual port interface between the buffer chip and the memory chips, and enabling the usage of 800-, 1,066-, 1,333-, and 1,600-MHz DDR3 memory chips, the POWER7 memory subsystem microarchitects greatly increased the raw DRAM bandwidth per buffer chip available to the channel scheduler. Similar to the POWER6 buffer chip, each POWER7 buffer chip port provides 8 bytes (72 bits) of DRAM bandwidth at DRAM frequency, adding up to 16 bytes for both ports. Additionally, the dual port interface enables twice as many memory chips to be connected to a POWER7 chip, thereby increasing the memory capacity. In certain system configurations, the memory subsystem will provide up to 256 GB of memory per POWER7 chip (up to 32 GB per core).

The scheduling rules for DDR3 memory create an additional challenge in arranging data. The POWER6 design spreads a 128-byte cache line across four 72-bit memory channels, yielding a 32-byte-wide data word protected by 32 bits of ECC. DDR2 scheduling rules are structured to optimally manage data in four-beat packets. Four beats of 32-byte-wide data words cleanly map to operations that manipulate 128-byte data granules. DDR3 scheduling rules are structured to optimally manage data in eight-beat packets. Therefore, the POWER7 design maps 128-byte cache lines into eight-beats of 16-byte-wide data words. That is, it spreads a given 128-byte-cache across only two 72-bit memory channels instead of four, yielding a 16-byte-wide data word protected by only 16 bits of ECC. In order to provide improved memory reliability, the POWER7 memory subsystem microarchitects developed a more complex, but significantly more robust, 64-byte marking code that employs 64 bits of ECC spread across four beats.

To support the wide variety of DDR3 frequencies efficiently, the 4 : 1 relationship between channel frequency and DRAM frequency that is supported by the POWER6 memory controller is augmented by a 6 : 1 relationship that is supported by the POWER7 memory controller. **Table 3** shows the raw peak bandwidth available at both the processor-to-buffer channel and at the buffer-to-DRAM interface for varying speed grades.

In order to capitalize on the increase in DRAM data bandwidth and channel bandwidth needed to satisfy the eight powerful cores, the memory subsystem microarchitects have invested significant resources into the buffering and reordering capabilities and significant sophistication into the scheduler. This provides a structure that accommodates the additional scheduling complexity resulting from the DDR3 rules, two channel-to-DRAM speed ratios, the reorganization of data into two channel pairs per controller,

Table 3 IBM POWER7 memory configurations.

<i>DDR3 DRAM frequency</i>	<i>Channel frequency</i>	<i>Speed ratio</i>	<i>Raw channel bandwidth</i>	<i>Raw DRAM data bandwidth</i>
800 MHz	4.8 GHz	6:1	135 GB/s	102 GB/s
1,066 MHz	6.4 GHz	6:1	180 GB/s	137 GB/s
1,333 MHz	5.3 GHz	4:1	149 GB/s	171 GB/s
1,600 MHz	6.4 GHz	4:1	180 GB/s	205 GB/s

the pipeline feedback effects of the CRC-based channel error management scheme, and the pipeline impacts of the multibyte 64-byte ECC marking code DRAM error management scheme. In certain system configurations, the POWER7 memory subsystem will provide in excess of 100 GB/s sustained memory bandwidth per chip (in excess of 12 GB/s per core).

The POWER7 memory subsystem includes accelerators for executing several new atomic memory update operations. These accelerators are architected to operate in concert with the intelligent network interface found in the cluster interconnect chip (described in the section on the cluster interconnect). The memory scheduler is further optimized to support these operations efficiently despite highly irregular access patterns. This capability enables large HPC clustered systems to provide global shared memory semantics without consuming processor resources local to the memory.

Similar to the POWER6 design, each memory controller is divided into two regions that operate at different frequencies. The *synchronous region* operates at the on-chip bus frequency, which, under ideal circumstances, is slightly faster than the 2 : 1 cycle (described in the section on the L2 cache). It manages the interaction with the on-chip coherence and data interconnect and participates as a point of coherence for the memory ranges it maps. It services reads and writes, arbitrates among conflicting requests, and manages coherence directory information. Each memory controller provides 64 general-purpose coherence management machines and 16 additional lightweight machines for managing data-zeroing operations. For read operations, numerous heuristics are employed to determine whether to access memory speculatively or to wait until a final coherence response notification has been received, essentially balancing access latency and energy usage.

Operations are passed from the synchronous region to the *asynchronous region*, which operates at one half of the channel frequency, or twice as fast as the DRAM when the channel is in 4 : 1 mode, and three times as fast as the DRAM when the channel is in 6 : 1 mode. It manages the interaction with the buffer chips and by extension, the DRAM chips. It includes the scheduler, which manages traffic through the channels and buffer chip, and schedules reads, writes, and maintenance operations such as refreshes and scrubs, carefully balancing and maximizing the

utilization of both the channel and the banking resources within the DRAM chips. At its disposal the scheduler has enough buffering to reorder up to 64 operations of up to 128 bytes each, providing a rich pool of scheduling opportunities (up to 16 KB per POWER7 chip). It manages the recovery from channel CRC errors and numerous types of 64-byte DRAM ECC errors while balancing several power control heuristics that limit memory access rates and place various portions of memory into different power management modes.

I/O subsystem

The POWER7 chip supports two integrated I/O controllers. They are built upon the same architectural foundation as the I/O controllers found in the POWER4, POWER5, and POWER6 chips. Each of the controllers supports a proprietary 4-byte off-chip read interface and a 4-byte off-chip write interface, thereby connecting the POWER7 chip to up to two I/O hub chips. These interfaces operate at frequencies ranging from 1.25 to 2.5 GHz, depending on the I/O hub chips to which they are connected. In addition to supporting the interrupt presentation function, pipelined I/O high-throughput mode, and advanced partial DMA write management pioneered in the POWER6 design, the POWER7 I/O controllers also support cache injection, which enables DMA write traffic to be written directly into a cache, instead of to memory.

By retaining the same I/O controller architecture, POWER7 systems are highly compatible with the POWER6 I/O ecosystem. However, by doubling the number of controllers per POWER7 chip, higher I/O throughput is enabled.

On-chip interconnect

As shown in Figure 1, the POWER7 chip contains eight cores, each with an associated L2 cache and local L3 region, two memory controllers, two I/O controllers, and a multichip SMP interconnect capable of scaling up to 32 POWER7 chips into a single large SMP system. Consequently, each POWER7 chip contains several coherence request, snoop, response, and notification interfaces and several data interfaces.

These are tied together by a high-bandwidth intelligent on-chip coherence and data interconnect that physically

occupies a horizontal equatorial trunk that runs across the center of the POWER7 chip. An emerging necessity, due to the multicore nature of the POWER7 design, the on-chip interconnect also manages the POWER7 capability to independently adjust individual core frequencies and accommodates the ensuing coherence and data flow variations. The on-chip interconnect operates at a frequency (called the on-chip bus frequency), which, under ideal circumstances, is slightly faster than the 2 : 1 cycle (described in the section on the L2 cache). The on-chip bus frequency is static and is consistent across all of the POWER7 chips in a system.

In order to continue to provide the high-scalability low-latency characteristics of earlier POWER server processors, the POWER7 processor utilizes a similar nonblocking-broadcast-based coherence-transport mechanism, based upon the same distributed management relaxed-order-optimized multiscope enablement provided by the POWER6 platform.

The on-chip coherence interconnect routes two sets (even and odd cache line) of coherence requests through the horizontal trunk, inward toward the even/odd arbitration logic located at the center of the chip. Up to one even request and one odd request may be granted each on-chip bus cycle. Once granted, the requests are broadcast within the chip on the even/odd snoop buses outward toward the left and right edges of the chip. Requests that will be routed to other chips are also sent to the multichip SMP interconnect (discussed in the multichip interconnect section) via a central vertical spine toward the top and bottom edges of the chip. Requests that have arrived from other chips are managed by the even/odd arbitration logic and broadcast to the snoop buses.

Coherence responses from the snoopers are routed inward along the horizontal trunk toward the even/odd coherence decision logic located at the center of the chip. For requests that have been routed to other chips, additional responses from the off-chip snoopers are fed into the coherence decision logic. Once a final coherence decision is made in response to a given request, a notification is broadcast within the chip on the even/odd notification buses outward from the center toward the left and right edges of the chip. Notifications that will be routed to other chips are also sent to the multichip SMP interconnect via the central vertical spine toward the top and bottom edges of the chip.

Because the coherence flow is nonblocking, the rate at which requests may be scheduled onto the snoop buses is restricted by the snoopers with the lowest possible snoop processing rate. The central coherence arbitration logic must insure that requests (whether sourced from the chip containing the slowest snoopers or from another chip in the system) do not overrun the slowest snoopers. To accommodate this, system firmware negotiates a “floor frequency.” As individual processor frequencies are adjusted

upward and downward, none will ever fall beneath the floor frequency. The coherence arbitration logic throttles the rate at which requests are granted to insure that a snooper operating at the floor frequency or higher can process all the requests.

The on-chip data interconnect consists of eight 16-byte buses that span the horizontal trunk. Four flow from left to right, and the other four flow from right to left. These buses are bracketed by memory controllers found at the left and right edges of the chip. They are divided into multiple segments, such that multiple 16-byte data packets may be pipelined within the multiple segments of the same bus at any given time. The buses operate at the on-chip bus frequency.

Each memory controller has two 16-byte on-ramps and two 16-byte off-ramps that provide access to the eight buses. Each core’s associated L2 cache and local L3 region share one 16-byte on-ramp/off-ramp pair, as does the pair of I/O controllers. The multichip data interconnect ports, found in the central vertical spines have a total of seven 16-byte on-ramp/off-ramp pairs. In total, there are twenty 16-byte on-ramps and twenty 16-byte off-ramps that provide access to and from the eight horizontal 16-byte trunk buses. Each ramp pair is associated with a bus segment. Note that a source-to-destination on-ramp/off-ramp route may consume only a subset of the segments in the horizontal trunk, depending upon the physical locations of the source and destination.

Data transfers are managed by centralized arbitration logic that takes into account source and destination locations, allocates available bus segments to plot one of several possible routes, allocates the on- and off-ramp resources, and manages destination data buffering resources. Note that since transfers may use only a subset of the segments in a given trunk bus, multiple noninterfering source-to-destination transfers may utilize the same horizontal trunk bus simultaneously. The arbitration logic must also account for the differing operating frequencies of the processor cores. For example, a source core operating at a lower frequency will send data via its on-ramp to the trunk buses at a slower rate. Likewise, a destination core operating at a lower frequency will consume data via its off-ramp from the trunk buses at a slower rate. To manage this, the arbitration logic controls speed-matching buffers in all of the on-ramps/off-ramps.

Multichip interconnect

The POWER7 system topology is built upon the structure that was developed for POWER6 systems. For standard commercial systems, up to 32 POWER7 chips may be connected to form a single 256-way SMP system.

Figure 11 depicts a first-level nodal structure, which combines up to four POWER7 chips. Each chip has four 10-B/s on-node SMP links associated with the vertical spine that emanates from the center of the chip toward the top edge.

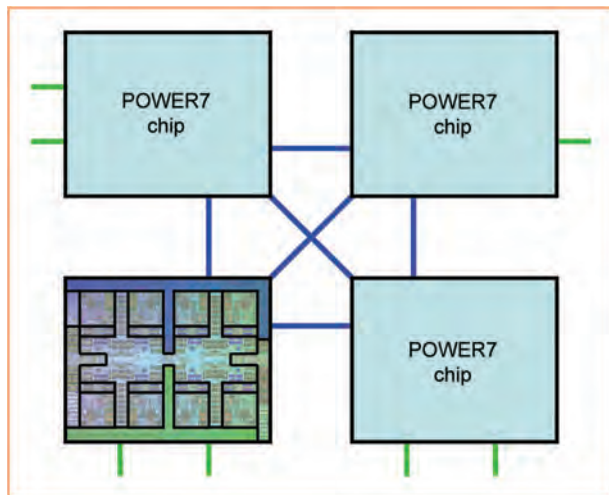


Figure 11

IBM POWER7 first-level nodal topology.

In a standard commercial system, three of these four SMP links are connected to each of the three other chips that comprise a four-chip node. In this manner, the four chips are fully connected.

A second-level system structure combines up to eight of the nodes. Each chip has two 10B/s off-node SMP links associated with the vertical spine that emanates from the center of the chip toward the bottom edge. As shown in **Figure 12**, in a standard commercial system, up to seven of the eight off-node SMP links (coming from the four POWER7 chips comprising a node) are connected to each of the seven other nodes that comprise an eight-node system.

Just as in POWER6 systems, the POWER7 SMP links are shared by coherence and data traffic and are protected by single-error-correct double-error-detect (SECDED Hamming code with additional parity) ECC. Unlike POWER6 systems, which enforce a strict 50% coherence to 50% data ratio or 33% coherence to 67% data ratio, the POWER7 SMP links enable a dynamic free-form allocation of coherence and data traffic, enabling higher effective utilization of the links. Additionally, to maximize SMP link bandwidth, the on- and off-node SMP links do not operate at the on-chip frequency. They are independently tuned, typically operating at speeds ranging from 2.5 to 3.3 GHz, depending upon system packaging characteristics, and provide increased flexibility over the POWER6 SMP links.

Despite the topological similarity, the POWER7 nonblocking multichip coherence transport had to be rearchitected due to the significant new challenges brought about by the per-core variable frequency capability. Based upon different system configurations, different processor core floor frequency settings, different SMP link frequencies, and the variation that arises from the

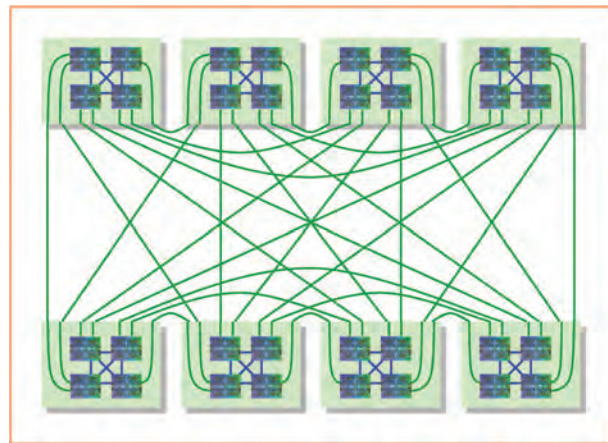


Figure 12

IBM POWER7 second-level system topology.

numerous asynchronous clock boundary crossings, the SMP coherence microarchitects developed a highly programmable sequencer that simultaneously balances priorities and flow rates for coherence requests initiated three chip-hops, two chip-hops, or one chip-hop away with those initiated from the same chip to achieve a nonblocking flow within prescribed latency variability and localized asynchronous crossing burst compression parameters.

From a performance perspective, the multiscope coherence capabilities introduced in the POWER6 design are critical for enabling balanced POWER7 scaling, all the way to 32 chips with 256 processor cores. Key ingredients for achieving robust multichip scalability are supporting a high volume of general coherence operations, as well as coherent atomic primitives, and being able to resolve them with low latency [8].

Since the POWER4 design, a broadcast-based distributed resolution coherence transport and rich cache coherence protocol have facilitated low latency resolution. In combination with the scheduling freedom enabled by the relaxed rules of the PowerPC storage architecture, they have also enabled extremely low overhead tracking structures, which are further exploited by the aggressive coherence reordering logic (also enabled by the relaxed storage architecture) to provide a high volume of coherence operations. For example, a 32-chip POWER7 system supports in excess of 20,000 concurrent coherence operations.

However, the volume of coherence traffic is limited by two physical constraints: 1) the bandwidth of the off-chip SMP links and 2) the rate (related to the “floor frequency”) at which the slowest snooper in the system can process coherence operations. While both of these have improved going from POWER6 chips to POWER7 chips, basic laws of physics and economics restrict the improvement to less

than $1.5\times$, which is less than the $4\times$ to $5\times$ improvement in system throughput.

Therefore, POWER7 systems heavily exploit the speculative localized scope coherence broadcast capability introduced in the POWER6 design. The localized regions make use of enhanced scope prediction heuristics to partition the coherence traffic, such that each region has full access to its SMP link and snoop coherence bandwidth. In cases where the speculation is successful, this has the effect of multiplying the link and snoop bandwidths by the number of regions. For example, dividing a large 256-way system into eight nodal regions has the effect (to the degree of successful speculation) of enabling each 32-way region to privately enjoy the SMP link and snoop bandwidth that would otherwise be shared across the whole system.

Cluster interconnect

For massive multipeta-FLOPs high-bandwidth clustered systems, a specialized cluster interconnect chip has been developed. The chip couples the POWER7 coherence and data interconnect seamlessly into a cluster interconnect. In this way, 32-way flat SMP systems are coupled to form 256-way SMP systems, which are interconnected to create a massive 512-K processor cluster.

The cluster interconnect chip has an on- and off-chip coherence and data interconnect that interoperates with the POWER7 chip and provides an on-board memory coherence directory (MCD) that expands upon the dual-scope capabilities provided by the POWER7 13-state coherence protocol. The MCD improves I/O subsystem throughput by qualifying and vectoring DMA reads and writes to a single target node, instead of broadcasting them to the entire SMP system. It also accelerates streaming memory write operations and reduces coherence scope misprediction latencies.

Each cluster interconnect chip incorporates up to three $16\times$ PCI Express** Gen2 I/O subsystem interfaces, a proprietary clustering interconnect interface, a high-bandwidth integrated switch and router, and a set of accelerators to process collective operations.

The cluster interconnect chip has four 10 B/s on-node SMP links to directly connect to four POWER7 chips. **Figure 13** illustrates how four POWER7 chips and one cluster interconnect chip are fully interconnected to form a flat 32-way SMP nodal construct with an integrated cluster interface. As described in **Table 4**, the connections between the four POWER7 chips are doubled, providing twice the nodal interconnect bandwidth as, compared with a standard system.

Each cluster interconnect chip has seven 10 B/s off-node SMP links, enabling each nodal construct to directly connect to seven other nodal constructs, forming an eight-node 256-way SMP system, primarily to facilitate coherent I/O sharing among the eight flat 32-way SMPs. The eight nodes

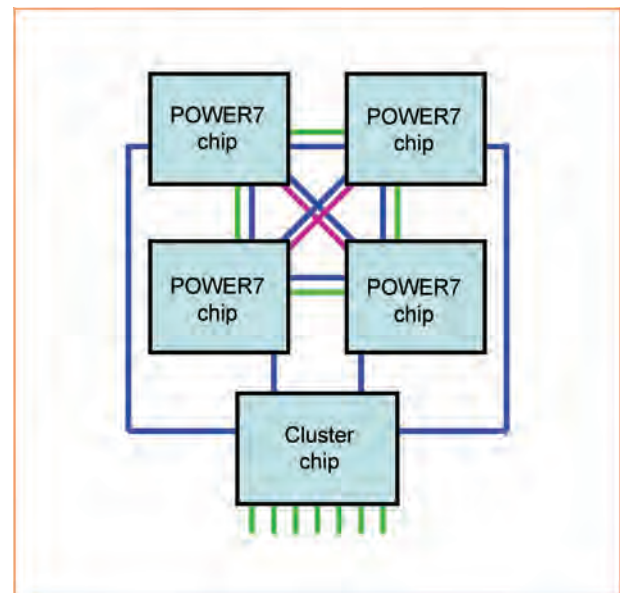


Figure 13

Cluster system first-level coherent nodal topology.

are fully connected in a manner similar to a standard 256-way system shown in Figure 12.

In addition to SMP coherence and data traffic, the seven off-node links also support cluster messaging traffic. Combined with 24 dedicated first-level optical cluster links, they enable a fully connected first-level cluster of 32×32 way SMP nodal constructs (or four 256-way SMP systems), which is called a SuperNode. Each SuperNode has 32 cluster interconnect chips and each cluster interconnect chip has 16 second-level optical cluster links. Therefore, each SuperNode has 512 available second-level optical cluster links. A massive 512-K processor cluster is composed of up to 512 SuperNodes, each directly connected to up to 511 neighboring SuperNodes via 511 of the 512 available second-level optical interconnect links.

Reliability, availability, and serviceability

POWER7 reliability and availability features are highlighted in **Figure 14**. POWER7 offers all the RAS features from the previous generation POWER6 processor such as core recovery, sparing, hot node add and repair, and redundant oscillators.

In POWER6, the core recovery mechanism was implemented through a separate recovery unit (RU), which stores previous architected states so that on error detection the processor core can be put back to an architected state that existed before the error condition occurred [13]. In POWER7, since the previous architected states are available in the register renaming mechanism implemented for

Table 4 IBM POWER7 standard system and cluster system interfaces.

Logical interface	Standard system	Cluster system
POWER7 to POWER7 nodal interconnect	Use three of four POWER7 on-node SMP links	Use three of four POWER7 on-node SMP links
POWER7 to POWER7 nodal interconnect 2× bandwidth expansion	n/a	Use two POWER7 off-node SMP links and both GX I/O links
POWER7 to cluster chip nodal interconnect	n/a	Use fourth POWER7 on-node SMP link
System interconnect	Use one or two POWER7 off-node SMP links	Use seventh cluster chip off-node SMP/cluster links
I/O subsystem attachment	Use POWER7 GX I/O links	Use cluster chip PCI interfaces
Memory channels	Use eight POWER7 channels	Use eight POWER7 channels

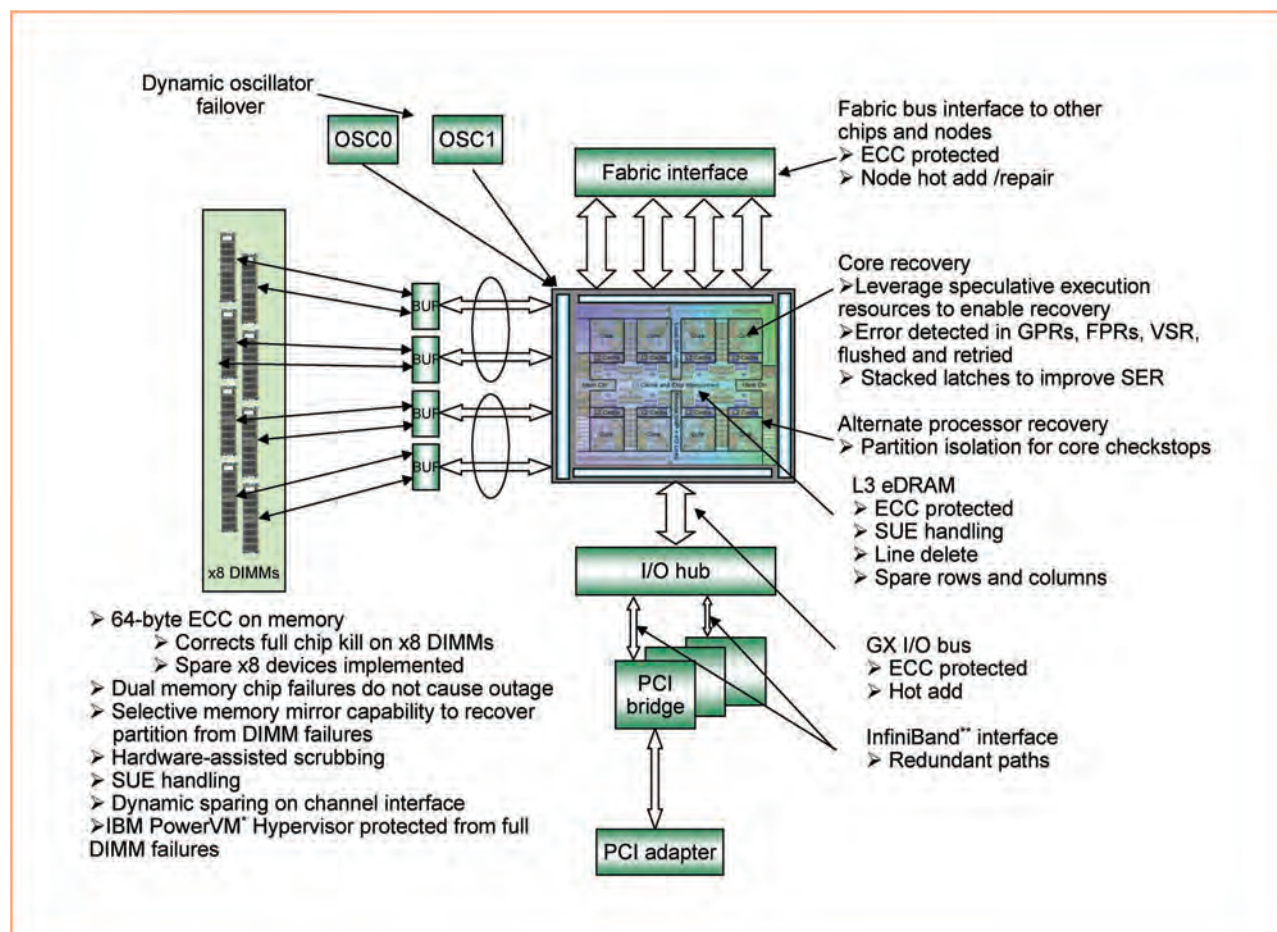


Figure 14

IBM POWER7 reliability and availability features.

out-of-order execution, no separate RU was required. When an error is detected and reported by a core unit, the POWER7 core quickly blocks all instruction completion, along with blocking all instruction fetch and dispatch. If the error condition is not severe enough to cause a checkpoint, the core initiates the recovery process. The recovery process flushes all the instructions in the pipeline for each thread to put the core in an architected state that existed sometime before the error condition occurred, fence the core from the rest of the system (L2 and nest), run an automatic built-in self-test to clear and test the core SRAM cells, reset each core unit to clean up any errors and reset the state machines, refresh the GPR and VSR files by initiating a state machine that does a read/correct/write operation to each entry in the register files to correct any single bit error through ECC correction mechanism, drop the fence to L2 and nest, and then restart instruction fetching and enable dispatch and completion.

To facilitate error detection and recovery in POWER7, the big register files (such as GPR and VSR) are ECC protected, whereas the smaller register files are protected through parity; all SRAM cells have error detection and recovery mechanisms. The I-cache is parity protected and recoverable. The floating-point pipelines implement residue checking mechanism, and numerous logic units implement additional control checking mechanism. In addition, POWER7 core uses RAS-hardened latches for various SPRs and core configuration latches.

The L1 D-cache is protected by byte parity. Hardware recovery is invoked on detection of a parity error while reading the L1 D-cache for a load instruction. The load instruction in error is not completed but rather flushed, the L1 D-cache contents are invalidated, and the instructions are refetched and reexecuted from the group of the load instruction in error. Additionally, when a persistent hard error is detected either in the L1 D-cache array or in its supporting directory or set predict array, a set delete mechanism is used to prohibit the offending set from being validated again. This allows the processor core to continue execution with slightly degraded performance until a maintenance action is performed.

Like the POWER6 design, the POWER7 L2 and L3 caches protect both data arrays and directory arrays with SECDED ECC, and the error management hardware supports autocorrection, autoinvalidation, autopurge capabilities, and physical cell deletion.

The POWER7 memory subsystem enjoys numerous enhancements in addition to those supported by the POWER6 memory subsystem. The 64-byte ECC algorithm allows a failing DRAM chip to be “killed” through marking so that the system can continue to run when an entire DRAM chip fails. In addition, certain system configurations provide spare DRAM chips so that a marked failed chip can be replaced by a spare DRAM chip, while the system is running. After the contents of the failed chip are “steered” to the spare chip, the ECC mark can be used to correct another chip kill error.

In certain system configurations, up to three complete DRAM chip failures per rank can be successfully corrected using the combination of chip sparing and marking.

The POWER7 memory channel design includes spare signal lanes that can be used to repair a failed channel lane dynamically, while the system is running. Hardware CRC counters and thresholding are used to cause a lane sparing action when an excessive CRC error rate is detected. Firmware also monitors each memory channel and can command a lane sparing action whenever a CRC rate is detected, which is not high enough to cause the hardware CRC threshold to be exceeded but is high enough to impact system performance.

In addition to memory channel CRC error detection, the POWER7 memory buffer chip contains logic to detect many internal error events. When a channel CRC or internal error event is detected, the internal buffer chip state is reset or repaired, and all memory operations in flight are retried. This provides a high level of fault tolerance for channel or buffer logic soft errors.

To provide an even higher level of reliability, certain system configurations provide a selective memory mirroring capability. In this mode, sections of memory are mirrored across memory channel pairs—memory writes are done to mirrored memory on both channel pairs, and mirrored memory reads are done from one of the channel pairs. If a hard memory ECC error or channel fail error occurs, correct data can be retrieved from the mirrored memory copy. The sections of memory to be mirrored can be dynamically selected and can coexist with sections of memory that are not mirrored. This flexibility allows users to select mirrored memory storage for critical applications and nonmirrored memory for less critical applications, thus providing high reliability without significantly increasing the total storage capacity required.

Summary

POWER7 continues the tradition of innovation in POWER line of processors. This seventh-generation chip adds balanced multicore design, eDRAM technology, and SMT4 to the POWER innovation portfolio. The POWER7 chip has four times as many cores and eight times as many threads, compared with POWER6 chip, as well as eight times as many FLOPs per cycle. The balanced design allows the processor to scale from a single socket low-end blade to a high-end enterprise system with 32 sockets, 256 cores, and 1024 threads. This new innovative design provides more than 4× performance increase per chip, compared with the previous generation POWER6 processor.

Acknowledgments

This paper is based upon work supported by the Defense Advanced Research Projects Agency under its Agreement No. HR0011-07-9-0002.

*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 PCI-SIG, InfiniBand Trade Association, or Sony Computer Entertainment Corporation in the United States, other countries, or both.

References

1. J. M. Tendler, J. S. Dodson, J. S. Fields, Jr., H. Le, and B. Sinharoy, "POWER4 system microarchitecture," *IBM J. Res. & Dev.*, vol. 46, no. 1, pp. 5–25, Jan. 2002.
2. B. Sinharoy, R. N. Kalla, J. M. Tendler, R. J. Eickemeyer, and J. B. Joyner, "POWER5 system microarchitecture," *IBM J. Res. & Dev.*, vol. 49, no. 4/5, pp. 505–521, Jul. 2005.
3. R. Kalla, B. Sinharoy, and J. Tendler, "IBM POWER5 chip: A dual-core multithreaded processor," *IEEE Micro*, vol. 24, no. 2, pp. 40–47, Mar./Apr. 2004.
4. H. Q. Le, W. J. Starke, J. S. Fields, F. P. O'Connell, D. Q. Nguyen, B. J. Ronchetti, W. M. Sauer, E. M. Schwarz, and M. T. Vaden, "IBM POWER6 microarchitecture," *IBM J. Res. & Dev.*, vol. 51, no. 6, pp. 639–662, Nov. 2007.
5. L. Eisen, J. J. W. Ward, III, H. Tast, N. Mäding, J. Leenstra, S. M. Mueller, C. Jacobi, J. Preiss, E. M. Schwarz, and S. R. Carlough, "IBM POWER6 accelerators: VMX and DFU," *IBM J. Res. & Dev.*, vol. 51, no. 7, pp. 663–684, Nov. 2007.
6. R. Kalla and B. Sinharoy, "POWER7: IBM's next generation POWER microprocessor," presented at the Hot Chips 21, Stanford, CA, Aug. 2009, DOI: doi.ieeecomputersociety.org/10.1109/MM.2010.2.
7. R. Kalla and B. Sinharoy, "POWER7: IBM's next-generation server processor," *IEEE Micro*, vol. 30, no. 2, pp. 7–15, Mar./Apr. 2010.
8. W. Starke, "POWER7: IBM's next generation, balanced POWER server chip," presented at the Hot Chips 21, Stanford, CA, Aug. 2009, DOI: doi.ieeecomputersociety.org/10.1109/MM.2010.2.
9. D. F. Wendel, R. Kalla, J. Warnock, R. Cargnoni, S. G. Chu, J. G. Clabes, D. Dreps, D. Hruscky, J. Friedrich, S. Islam, J. Kahle, J. Leenstra, G. Mittal, J. Paredes, J. Pille, P. J. Restle, B. Sinharoy, G. Smith, W. J. Starke, S. Taylor, A. J. Van Norstrand, S. Weitzel, P. G. Williams, and V. Zyuban, "POWER7, a highly parallel, scalable multi-core high end server processor," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 145–161, Jan. 2011.
10. POWER ISA, ver. 2.06. [Online]. Available: http://www.power.org/resources/downloads/PowerISA_V2.06B_V2_PUBLIC.pdf
11. S. S. Iyer, J. E. Barth, Jr., P. C. Parries, J. P. Norum, J. P. Rice, L. R. Logan, and D. Hoyniak, "Embedded DRAM: Technology platform for the blue gene/L chip," *IBM J. Res. & Dev.*, vol. 49, no. 2/3, pp. 333–350, Mar. 2005.
12. S. S. Iyer, G. Freeman, C. Brodsky, A. I. Chou, D. Corliss, S. H. Jain, N. Lustig, V. McGahay, S. Narasimha, J. Norum, K. A. Nummy, P. Parries, S. Sankaran, C. D. Sheraw, P. R. Varanasi, G. Wang, M. E. Weybright, X. Yu, E. Crabbe, and P. Agnello, "45-nm silicon-on-insulator CMOS technology integrating embedded DRAM for high-performance server and ASIC applications," *IBM J. Res. & Dev.*, vol. 55, no. 3, pp. 5:1–5:14, 2011.
13. M. J. Mack, W. M. Sauer, S. B. Swaney, and B. G. Mealey, "IBM POWER6 reliability," *IBM J. Res. & Dev.*, vol. 51, no. 6, pp. 763–774, Nov. 2007.

Received November 17, 2010; accepted for publication February 14, 2011

Balaram Sinharoy IBM Systems and Technology Group, Poughkeepsie, NY 12601 USA (balaram@us.ibm.com). Dr. Sinharoy is an IBM Distinguished Engineer and the Chief Architect of the IBM POWER7 processor. Before POWER7, he was the Chief Architect for the POWER5 processor. He has published numerous articles and received more than 50 patents in the area of computer architecture, with many more patents pending. Dr. Sinharoy also received several IBM corporate awards for his work in different generations of POWER microprocessors. He is an IBM Master Inventor and an IEEE Fellow.

Ron Kalla IBM Systems and Technology Group, Austin, TX 78758 USA (rkalla@us.ibm.com). Mr. Kalla is the Chief Engineer for IBM POWER7. He has 25 years of processor design experience. He has worked on processors for IBM S/370, M68000, iSeries*, and pSeries* machines. He holds numerous patents on processor architecture. He also has an extensive background in post silicon hardware bring-up and verification. He has 30 issued U.S. patents and is an IBM Master Inventor.

William J. Starke IBM Systems and Technology Group, Austin, TX 78758 USA (wstarke@us.ibm.com). Mr. Starke joined IBM in 1990 after graduating from Michigan Technological University with a B.S. degree in computer science. He is a Senior Technical Staff Member in the POWER development team of the Systems and Technology Group. After several years of cache hierarchy and symmetric multiprocessor hardware performance analysis for both IBM mainframe and POWER server development programs, he transitioned to logic design and microarchitecture development, working initially on the POWER4 and POWER5 programs. Mr. Starke led the development of the POWER6 cache hierarchy and SMP interconnect and served as the Chief Architect for the POWER7 storage hierarchy (i.e., cache hierarchy, SMP interconnect, memory subsystem, and I/O subsystem). He currently serves in the Storage Hierarchy Chief Architect role for the POWER8* program. A prolific innovator, Mr. Starke holds more than 100 issued U.S. patents, in addition to several currently pending.

Hung Q. Le IBM Systems and Technology Group, Austin, TX 78758 USA (hung@us.ibm.com). Mr. Le joined IBM in 1979 after graduating from Clarkson University with a B.S. degree in Electrical and Computer Engineering. He is an IBM Fellow in the POWER development team of the Systems and Technology Group. He worked on the development of several IBM mainframe and POWER/PowerPC processors and has been contributing to the technical advancement of IBM processor technology. His technical interests are in the field of processor design involving multithreading, out of order design, thread level parallelism for multi core. He currently holds more than 90 U.S. issued patents.

Robert Cargnoni IBM Systems and Technology Group, Austin, TX 78758 USA (cargnoni@us.ibm.com). Mr. Cargnoni received a B.S. and M.S. degrees in electrical engineering from the University of Illinois at Urbana-Champaign. He is the Chief Engineer for the IBM POWER7 cache hierarchy, coherence protocol, SMP interconnect, and memory and I/O subsystems. He held key leadership positions in the POWER4 and POWER5 programs as well.

James A. Van Norstrand IBM Systems and Technology Group, Austin, TX 78758 USA (njvan@us.ibm.com). Mr. Van Norstrand graduated from Syracuse University in 1982 with a B.S.E.E. degree. He is a Senior Technical Staff Member in the POWER development team. He was the unit lead for the instruction fetch unit in POWER7. Before POWER7, he was the core lead on the Cell Broadband Engine** chip, POWER4 lab manager, and zSeries designer for the instruction fetch unit.

Bruce J. Ronchetti *IBM Systems and Technology Group, Austin, TX 78758 USA (ronchetti@us.ibm.com).* Mr. Ronchetti joined IBM in 1979 after receiving a B.S. degree in electrical engineering from Lafayette College. He is a Senior Technical Staff Member in the POWER system development area. For the past 12 years, he has focused on processor core microarchitecture development, particularly in load and store units.

Jeffrey Stuecheli *IBM Systems and Technology Group, Austin, TX 78758 USA (wstarke@us.ibm.com).* Mr. Stuecheli has a B.S. degree in computer engineering in 1997 and an M.S. degree in 2004, both from the University of Texas at Austin. He expects to receive his Ph.D. degree in May 2011. He joined IBM in 1997 and is a Senior Engineer in the POWER development team of the IBM Systems and Technology Group. His current focus is on enhancements to the memory subsystem targeting increased performance.

Jens Leenstra *IBM Systems and Technology Group, Boeblingen D-71032, Germany (leenstra@de.ibm.com).* Dr. Leenstra is an IBM Senior Technical Staff Member and the lead for the IBM POWER7 vector and scalar unit. He worked on the design and verification of I/O chips, multiprocessor system verification of the IBM S/390 G2 and G3 mainframe computers, the Cell Broadband Engine processor synergistic processing elements, and POWER6 processor VMX unit. He has 20 issued patents and is an IBM Master Inventor.

Guy L. Guthrie *IBM Systems and Technology Group, Austin, TX 78758 USA (gguthrie@us.ibm.com).* Mr. Guthrie received a B.S. degree in electrical engineering from Ohio State University. He is a Senior Technical Staff Member in the POWER development team of the Systems and Technology Group and is an architect for the IBM POWER7 cache hierarchy, coherence protocol, SMP interconnect, and memory and I/O subsystems. He served in a similar role for POWER4, POWER5, POWER6 programs as well. Prior to that, he worked as a hardware development engineer on several PCI Host Bridge designs and also worked in the IBM Federal Systems Division on a number of IBM Signal Processor Development programs. He is an IBM Master Inventor and holds more than 100 issued U.S. patents.

Dung Q. Nguyen *IBM Systems and Technology Group, Austin, TX 78758 USA (dqnguyen@us.ibm.com).* Mr. Nguyen joined IBM in 1986 after graduating from the University of Michigan with an M.S. degree in materials engineering. He is a Senior Engineer in the POWER development team of the Systems and Technology Group. He has worked on the development of several processors, including POWER3*, POWER4, POWER5, POWER6, and POWER7. He is currently working on the POWER8 microprocessor. Mr. Nguyen technical interests are in the field of processor design involving instruction sequencing and multithreading.

Bart Blaner *IBM Systems and Technology Group, Essex Junction, VT 05452 USA (blaner@us.ibm.com).* Mr. Blaner earned a B.S.E.E. degree from Clarkson University. He is a Senior Technical Staff Member in the POWER development team of the Systems and Technology Group. He joined IBM in 1984 and has held a variety of design and leadership positions in processor and ASIC development. In the past several years, he has focused on POWER processor microarchitecture and design, leading the POWER7 fixed-point unit implementation. Recently his attention has turned to design and implementation of hardware accelerators for compute-intensive algorithms. He is a Senior Member of the IEEE and holds more than 30 patents.

Charles F. Marino *IBM Systems and Technology Group, Austin, TX 78758 USA (marinoc@us.ibm.com).* Mr. Marino received his B.S. degree in electrical and computer engineering from Carnegie-Mellon University. He is a Senior Engineer in the IBM Systems and Technology Group. In 1984, he joined IBM in Owego, New York. Mr. Marino is currently the fabric team lead for the IBM POWER7 servers.

Eric Retter *IBM Systems and Technology Group, Austin, TX 78758 USA (retter@us.ibm.com).* Mr. Retter joined IBM in 1985 after graduating from the Pennsylvania State University with a B.S. degree in engineering science. He is a Senior Engineer in the POWER development team of the Systems and Technology Group. He began his career in the IBM Federal Systems Division in Owego, New York, working as a logic designer and team lead on multiple military avionics computer programs. In 1997 he joined the IBM Digital Video Products Group in Endicott, New York, as a logic designer for digital set-top box system-on-a-chip designs, and later served as Program Technical Lead for the IBM Digital Video Products Group's low-cost set-top box chip. In 2003 Mr. Retter transferred to the IBM Systems and Technology Group in Austin, Texas, working initially as a logic designer on the POWER6 memory controller, and later as logic team lead for the POWER6 and POWER7 memory controller designs. He currently serves in the memory controller Logic Team Lead role for the POWER8 program.

Phil Williams *IBM Systems and Technology Group, Austin, TX 78758 USA (willamp@us.ibm.com).* Mr. Williams is a graduate of Missouri Institute of Technology. He is a Senior Engineer and currently the Lead Designer for embedded DRAM cache on POWER processors. He has worked in a variety of areas on processors in his 30 years at IBM and holds numerous patents in the areas of branch instruction processing, caches, and interconnects.

IBM POWER7 systems

R. X. Arroyo
R. J. Harrington
S. P. Hartman
T. Nguyen

This paper describes the system architectures and designs of the IBM POWER7® servers. From the smallest single-processor socket blade to the largest 32-processor-socket 256-core enterprise rack server, each system is designed to fully exploit the performance and the scalability of the POWER7 processor. This paper describes the enhancements made to the memory and input/output subsystems to achieve balanced and scalable designs, the changes made to the power and cooling circuitry to manage energy consumption and power dissipation, and the enhancements made to reliability, availability, and serviceability. These enhancements enable the POWER7 processor-based servers to achieve significant increases in the performance density and the performance per watt, as compared with the predecessor POWER6® processor-based servers.

Introduction

As the name implies, the IBM POWER7* processor is the seventh generation of the reduced instruction set computing (RISC) processor technology utilized in IBM's UNIX**-based systems. The first generation was introduced in 1990 with the development of the IBM RISC system RS/6000* family of systems [1]. Since then, IBM has continued to introduce new processor and system technologies through seven generations of the POWER* processor technology, culminating with the 2010 introduction of the POWER7 processor.

The predecessor POWER6* systems were introduced in 2007, and at the time of their introduction, they extended IBM's position in the UNIX-based server market with the integration of the POWER6 ultrahigh-frequency central processing unit (CPU) design [2]. IBM offered a wide range of system designs that were based on POWER6 that ranged from a scale-out two-core blade offering to a high-end scale-up 64-core enterprise rack server offering. These systems have been now enhanced, modified, and redesigned for POWER7 to meet the additional memory and input/output (I/O) capacity and bandwidths required for scalable balanced systems. The POWER7 systems now range from a four-core blade system to a 256-core high-end enterprise server. In this paper, we describe the various POWER7 system designs and architectures. **Table 1** provides an overview of the IBM POWER7 systems. The basic architecture for all of the POWER7 systems is the

same. However, there are workload and scalability differences across the systems that drive different system designs, as shown in Table 1.

System technologies

Here, we describe the base technologies that are common for all of the system designs.

IBM POWER7 processor

The POWER7 processor is a high-performance processor that can scale up to 256 cores [3]. Each processor die contains eight powerful computing cores, 32 MB of level-3 (L3) shared cache, a highly scalable processor switch fabric that connects up to 32 sockets, two high-bandwidth double-data-rate 3 (DDR3) memory controllers, and an I/O controller that provides two high-speed I/O mezzanine buses called *GX buses*. There are two sets of fabric buses for the POWER7 processor: the nodal fabric interconnect buses named W, X, Y, and Z; and the symmetric multiprocessing (SMP) buses named A and B buses. The nodal fabric buses are used for interconnections of up to four processor sockets with a direct fabric bus connection to every other processor. Only three of the four nodal buses are required for this topology, and any three of the four buses can be used. A fourth nodal fabric bus is provided for future use. The A and B SMP fabric buses are then used to connect multiple nodes together with at least one A or B bus directly connected between each of the nodes in a system.

The highly flexible design of the POWER7 processor enables a wide range of system designs while enabling

Digital Object Identifier: 10.1147/JRD.2011.2131610

©Copyright 2011 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied by any means or distributed royalty free without further permission by computer-based and other information-service systems. Permission to republish any other portion of this paper must be obtained from the Editor.

0018-8646/11/\$5.00 © 2011 IBM

Table 1 IBM POWER7 systems overview. (Note: In rack height descriptions, “U” refers to unit of measurement for rack height; 1U = 1.75 inches.)

<i>System</i>	<i>PS700</i>	<i>PS701</i>	<i>PS702</i>	<i>710</i>	<i>720</i>	<i>730</i>	<i>740</i>	<i>750</i>	<i>755</i>	<i>770</i>	<i>780</i>	<i>795</i>
<i>System enclosure</i>	IBM BladeCenter			19-in. rack								24-in. rack
<i>Rack height</i>	1 blade	1 blade	2 blades	2U	4U	2U	4U	4U	4U	16U	16U	42U
<i>Second form factor</i>				Tower		Tower						
<i>CPU sockets</i>	1	1	2	1	1	2	2	4	4	8	8	32
<i>CPU cores</i>	4	8	16	8	8	16	16	32	32	64	64	256
<i>DIMM slots</i>	8	16	32	8	16	16	32	32	32	64	64	256
<i>Maximum memory</i>	64 GB	128 GB	256 GB	64 GB	128 GB	128 GB	256 GB	512 GB	256 GB	2 TB	2 TB	8 TB
<i>PCI slots</i>	2	2	4	4	8	4	8	5	5	24	24	NA
<i>GX slots</i>	NA	NA	NA	1	1	2	2	2	2	8	8	32
<i>2.5-in. disk bays</i>	2	1	2	6	8	6	8	8	8	24	24	NA
<i>I/O expansion</i>	NA	NA	NA	No	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes

several price and performance options within a single system. For example, most of the systems offer a variety of processor module options that differ in the number of cores that are active. In addition to the eight core modules, there are six core processor modules and four core processor modules. These modules all use the same processor die, except that two or four of the processor cores are permanently disabled.

The flexibility of the POWER7 processor design also enables optimizations for smaller systems that are four processor sockets or less and do not require the full set of fabric buses to interconnect the processors. In addition, these systems are designed for smaller enclosures that limit the number of dual inline memory module (DIMM) slots that can be included in the design, and therefore, not all eight memory channels can be implemented in the smaller system designs. The POWER7 processor is designed to operate in these configurations, and as a result, there are two single-chip module (SCM) packages designed for the POWER7 processor chip, a full-function SCM that exposes all of the capabilities of the POWER7 processor die, and an entry SCM that is optimized for smaller systems where the cost and the size are critical factors.

The entry SCM is constructed out of a low-cost organic material. To minimize the size and the pin count of this package, the entry SCM does not contain any pins for the SMP fabric A and B buses. Three nodal fabric buses (W, Y, and Z) are provided, and just 4 bytes of each bus are implemented. Finally, this module exposes only four of the eight memory channels. The result is a 50 mm × 50 mm organic module with 2,294 total pins that supports up to 190 W of power dissipation.

In the larger systems that contain 8–32 sockets, the scalability and the performance are critical factors. The

full-function SCM of the POWER7 processor contains both the nodal fabric buses and the SMP fabric buses, and to ensure the scalability in these large system configurations, all 8 bytes of the fabric and interconnect buses are implemented. For maximum performance, both memory controllers are used in these system configurations, and therefore, all eight memory channels are externalized in this module. Finally, a glass ceramic material construction for this module provides support for up to 250 W, which allows these systems to operate at the highest CPU frequencies. The size of the module is 61 mm × 61 mm, and it contains 3,363 pins. **Figure 1** summarizes the I/O for the two POWER7 modules designed for these systems.

Another enhancement to the flexibility of the POWER7 processor design is the separate clocking domain for the processor core. This enables the processor core to run asynchronously to the fabric buses and the GX I/O bus (the POWER6 core was already asynchronous to the memory bus). In POWER6, the fabric and I/O buses operated at a fixed ratio of the processor core frequency. As the processor core frequency changed, the I/O bus frequency proportionally changed, and since the IBM I/O hub controller was not designed for a wide range of frequency operation, this limited the possible amount of dynamic change to the processor frequency. The asynchronous design allows the systems to dynamically operate the POWER7 cores as low as 50% of the nominal frequency to achieve greater levels of energy savings.

The second benefit of this asynchronous design feature is that it allows the CPU to be “overclocked,” i.e., to operate at a frequency that is higher than the nominal frequency. Additional power and cooling associated with these higher frequencies require enhancements to the system designs.

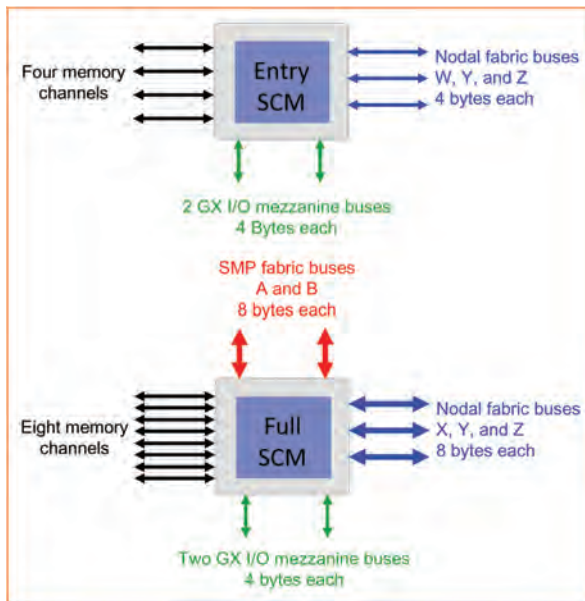


Figure 1

IBM POWER7 SCMs.

These enhancements have been designed into all POWER7 systems, with some systems capable of supporting as much as a 9% increase in the frequency above nominal. With the POWER7 asynchronous design, a significant increase in the dynamic range of frequencies (–50% to +9%) is now available for the EnergyScale* dynamic power saver mode [4] to not only provide greater energy savings but also have the capability to satisfy peak workload requirements.

Thermal and power management

As mentioned in the previous section, there are several enhancements that were implemented in the POWER7 system designs to accommodate the POWER7 processor and support several improvements to the EnergyScale power and thermal management utilities. All of the POWER7 systems use a dedicated microprocessor to obtain continuous real-time monitoring of processor utilization, subsystem power consumption, and total system power consumption, as well as processor and memory thermal monitoring. This microprocessor, called the temperature and power management device (TPMD) is a 32-bit RISC microprocessor from the Renesas SuperH* family of processors. This microprocessor is connected to each processor in the system through an industry-standard serial two-wire (I^2C) [5] bus that it uses to query the utilization of each processor in the system. The TPMD also contains two analog-to-digital converters with eight channels, each of which is used to monitor power across several subsystems

such as each CPU, memory, I/O subsystem, storage subsystem, fans, and total current draw of the system.

In addition to CPU utilization, the TPMD uses the I^2C buses to monitor the POWER7 processor thermal temperatures through digital temperature sensors located on the processor die. These temperatures and the temperatures on the DIMMs and the I/O are gathered by the system service processor and are used to control the speed of the system fans to provide closed-loop temperature feedback for the fan control logic. As the device temperatures rise, the fan speeds are increased, and as the device temperatures fall, the fan speeds are decreased.

Memory buffer chip

As previously mentioned, the POWER7 processor chip contains two DDR3 memory controllers that drive four external high-speed memory channels each. As in POWER6, a memory buffer chip is implemented between the processor and the main memory to provide a wide high-bandwidth interface to the dynamic random-access memory (DRAM). Each memory channel from the processor is connected to an IBM DDR3 memory buffer chip that contains several enhancements over the POWER6 memory buffer chip. First, the memory interface side of the buffer chip has an additional memory port for a total of two full 8-byte synchronous DRAM (SDRAM) memory ports, with each port supporting up to eight ranks of DDR3 1,066-MHz SDRAMs. Correspondingly, the processor interface side of the buffer chip has been enhanced to support a high-speed differential-ended memory channel running at six times the memory frequency 6.4-Gb/s transfer rate. Each memory channel consists of two unidirectional links. A downstream link is used to transmit the write data and commands to the memory. It consists of 13 data lanes, 2 spare lanes, and 1 clock. The second link is the upstream link, which is used to read data and responses from the memory and the buffer. It consists of 20 data lanes, 2 spare lanes, and 1 clock.

Two types of memory buffer topologies are implemented in POWER7 systems. One topology places the memory buffer on the planar (or a riser) and allows the use of one or two industry-standard registered DIMMs per memory port. This topology is primarily used in blades and entry systems to leverage commodity DIMM technologies widely available in the industry.

The second topology places the memory buffer on an IBM unique fully buffered 276-pin DIMM. This fully buffered DIMM is 98 mm tall and contains one memory buffer, up to 16 ranks of DDR3 SDRAMs, and two thermal sensors to provide a level of memory density that cannot be achieved with industry-standard memory DIMMs. This is critical for enterprise-class systems that require large amounts of memory for highly virtualized environments and for maximum commercial transaction performance.

I/O hub controller

The POWER7 processor chip contains an I/O controller that has the same architecture as the I/O controller designed in POWER6, POWER5* [6], and POWER4* [7]. The I/O controller in POWER7, however, has added a second GX I/O bus that doubles the I/O bandwidth capability per processor socket. Each bus is a single-ended point-to-point unidirectional bus that contains 4 bytes for read and 4 bytes for write. This bus has a maximum speed of 2.5 GHz and asynchronously operates to the CPU core frequency. This GX I/O bus connects to an IBM designed I/O hub controller chip, which is a multifunctional bridge chip that generates several PCI-X** buses and PCI Express** (PCIe**) buses to attach I/O adapters and components. It also contains an Ethernet controller and generates a pass-through I/O bus for additional I/O expansion.

The I/O hub controller chip provides four $\times 8$ PCIe ports that are version 1.1 compliant [8] and are used in the POWER7 systems to connect to integrated I/O components such as a disk controller or routed to industry-standard PCIe slots. Each $\times 8$ port supports a 2.0-GB/s bandwidth in each direction (4-GB/s concurrent). Each of the four PCIe ports can run in $\times 8$, $\times 4$, $\times 2$, or $\times 1$ mode.

There are four separate PCI-X buses that POWER7 systems route to integrated I/O components or route to an industry-standard PCI-X slot. Two of the four PCI-X buses are PCI-X 2.0 compliant [9] running at a maximum speed of 266 MHz, and two are PCI-X 1.0 compliant running at a maximum speed of 133 MHz. Each of the two PCI-X 2.0 buses is 64 bits in width, whereas one PCI-X 1.0 bus is 64 bits in width, and the other is 32 bits in width. All four buses are highly configurable and can run in the PCI mode. The PCI-X 2.0 buses can run in the PCI-X 1.0 mode as well.

The I/O hub controller chip provides two 10-Gb XAUI Ethernet ports that are compliant with the IEEE standard 802.3 [10]. Each 10-Gb port can be individually programmed to function as two 1-Gb Ethernet ports. This programmability must match the Ethernet physical transceivers implemented in each system; therefore, most POWER7 systems implement a pluggable daughter card for the Ethernet physical transceivers. This allows the system to support both 10- and 1-Gb ports based on the type of pluggable daughter card that is installed. In addition, the I/O hub controller has implemented a I/O virtualization hardware that allows the physical Ethernet ports to be directly shared by multiple logical partitions running on the system. This virtualization hardware eliminates the need for multiple partitions to go through a dedicated virtual I/O server partition to access the Ethernet function. The I/O hub controller supports 16 logical ports (partitions) per 10-Gb physical port.

System architectures

The systems listed in Table 1 can be categorized into four basic system architectures, i.e., blades, 2- to

4-socket systems, 8-socket systems, and a 32-socket system.

Blades

The PS700, PS701, and PS702 blades are designed to comply with the form factor and power requirements of the IBM family of BladeCenter* systems. Specifically, these blades plug into the IBM BladeCenter H, BladeCenter HT, and BladeCenter S chassis, with the PS700 blade also able to plug into BladeCenter E and BladeCenter T. These BladeCenter systems differ in the amount of power, cooling, and number of blades supported, but each BladeCenter supports the same blade form factor, which is 9.65 inches (245 mm) high by 1.14 inches (29 mm) wide by 17.55 inches (445 mm) deep. Each blade is a complete system design with processors, memory, and I/O, with the bulk power supplies and fans for all of the blades located in the BladeCenter chassis. The IBM BladeCenter provides the densest packaging of the IBM POWER7 systems and the densest scale-out solution.

The architecture and the design of the blade revolve around how to fully optimize the system design in the fixed form factor and the fixed power allocation of the blade. For POWER7, one of the key decisions in the architecture of the blade design was to reduce the number of CPU sockets from the two sockets (POWER6 blade) to a single-CPU-socket implementation. This design point still results in the doubling of the number of processor cores, as compared with POWER6, but more importantly, it results in a much more balanced system design for the POWER7 processor. For example, in the POWER6 blade, each of the two CPU sockets dissipated a maximum of 110 W each, for a total of 220 W. With a single-socket POWER7 CPU blade, the total CPU power is lowered to 150 W. This amount of power savings is now allocated to the additional memory capacity and bandwidth that is required for maximizing the POWER7 processor performance. The POWER7 blades contain twice the number of DIMM slots as the POWER6 blades since less board space is needed for the CPU sockets.

IBM offers three highly derivative blade designs that are based on the architecture described above. This architecture is shown in **Figure 2**. The PS700 is a four-core blade, the PS701 is an eight-core blade, and the PS702 is a 16-core double blade. The PS702 occupies two adjacent blade slots in the IBM BladeCenter systems. The two CPU sockets in the PS702 are connected through one 4-byte nodal fabric bus (W bus) running at 2.9 Gb/s. Each processor has four memory channels running at a rate of 6.4 Gb/s and up to 16 DDR3 very-low-profile DIMMs except for the PS700 blade, which has two memory channels and eight DIMMs per socket. These blades provide up to 16 GB per core.

The I/O subsystem within a blade is driven by the I/O hub controller, which generates PCIe buses and PCI-X buses

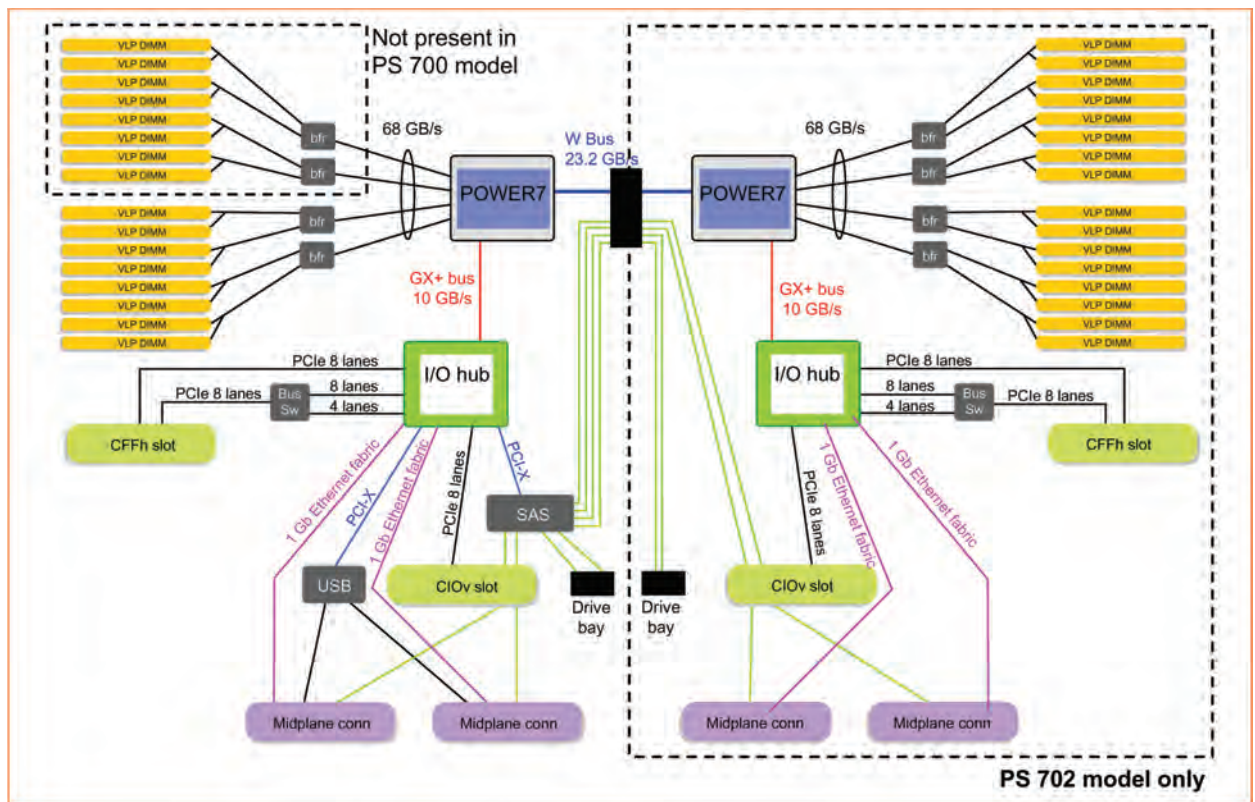


Figure 2

Blade architecture.

to the internal-blade I/O slots, a disk serial-attached small-computer serial interface (SAS) controller, and a Universal Serial Bus (USB) device. PS700 and PS701 support two internal-blade I/O slots, a PCIe CIOv (i.e., card I/O, vertically mounted) expansion slot, and a PCIe CFFh (i.e., combo form factor, horizontally mounted) high-speed expansion slot. With its two-blade structure, PS702 supports two of each type of the I/O slot for a total of four internal-blade I/O slots. PS700 is able to package two 2.5-inch disk drives in each blade, PS701 contains one 2.5-inch disk drive, and PS702 supports two 2.5-inch disk drives (one in each blade).

Each blade plugs into a BladeCenter chassis midplane through dual connectors. Each connector contains a base 1-Gb Ethernet fabric interconnect, a USB channel for keyboard mouse, and SAS lanes to access the storage contained in the BladeCenter chassis.

Two- and four-socket entry systems

The 710, 720, 730, 740, 750, and 755 models make up the two- and four-socket POWER7 entry systems, which are designed to be installed in standard 19-inch racks. Only the

720 and 740 models offer a second system packaging option, which is a standalone tower enclosure. These models can be subdivided into two basic designs, i.e., two-socket systems (710, 720, 730, and 740) and four-socket systems (750 and 755). Each of these designs has many architectural similarities, but there are variations that result in two distinct system designs that are optimized for different market segments in the entry server space.

Four-socket system architecture

There are two highly derivative four-socket system offerings, the 750 and the 755 systems that have the same architecture and are differentiated only by the processor frequencies supported by each system. These four-socket systems were the first to be released with the POWER7 technology. This was accomplished by significantly leveraging the existing POWER6 four-socket system. The challenge was to modify the existing POWER6 four-socket system to support the POWER7 CPU and its enhanced DDR3 memory subsystem without changing the basic mechanical structure of the system and without changing the I/O subsystem.

The POWER6 four-socket system is structured such that there are four CPU cards that contain a single processor module and its locally attached memory. The four CPU cards plug vertically down onto an I/O backplane that is positioned at the bottom of the drawer chassis. The I/O backplane contains all of the integrated I/O of the system. Since the GX I/O bus remained the same between POWER6 and POWER7, it was possible to redesign the CPU cards to support the POWER7 processor and not modify the I/O backplane. The CPU cards contain a POWER7 CPU module (the entry SCM module), eight industry-standard DDR3 memory DIMM slots, four POWER7 memory buffer modules, and new enhanced direct-current (dc)-to-dc regulators. The memory channel speed was increased to 6.4 Gb/s, and the DDR3 memory interface was designed for a 1,066-MHz data rate. The dc-to-dc regulators were enhanced to support the POWER7 overclocking frequencies and the additional power required for the DDR3 memory and to provide accurate current-draw monitoring capability and thermal monitoring capability for advanced power and thermal management. This was accomplished within the same card dimensions as the POWER6 CPU card, alleviating the need for any structural mechanical changes or changes to the I/O backplane.

The second major challenge in this design was to address the additional bulk power required by the new POWER7 CPU cards. The two power supplies in the four-socket system are rated at 1,725 W each. A minimum of one power supply is required for the system to operate, and the second power supply is for redundancy. In the POWER6 four-socket design, the worst case total system power was less than 1,725 W, but for the POWER7 system, the worst case total system power is 1,800 W, which is more than what a single power supply can provide. The additional power is required to support the POWER7 “overclocking” mode and the additional power for the memory subsystem with its higher utilization and a larger 16-GB DDR3 1,066-MHz memory DIMM.

With two power supplies present in a system, the total system load is split across the two power supplies, and thus, each power supply delivers approximately 900 W for a 1,800-W system load, which is well below its maximum dc output rating of 1,725 W. To take advantage of this dual power-supply capability, the system must be able to immediately react to a failed power-supply condition and throttle the system to under 1,725 W within 100 ms to avoid an overcurrent condition. This is accomplished with the implementation of the enhanced energy management logic and with changes to the service processor code in the system to immediately throttle the CPU and the memory performance once a power-supply failure is detected. This methodology eliminated the need to grow the power supplies, which would have required several structural changes to the system.

Figure 3 displays the resulting system design of the four-socket system. The four sockets are connected together with the 4-byte WYZ nodal fabric buses running at 2.9 Gb/s. Each processor has four memory channels running at a rate of 6.4 Gb/s and up to eight DDR3 DIMMs. This system supports six core partially good modules, in addition to the eight-core processor modules. The maximum amount of memory supported is 21.3 GB per core for the six-core processor module or 16 GB per core for the eight-core processor module.

The I/O subsystem is connected to the processors through two GX I/O buses. One GX I/O bus from the first CPU card is used to connect to the I/O hub controller that drives two PCI-X DDR slots, three PCIe $\times 8$ slots, and the integrated I/O such as USB, Ethernet, and a SAS disk controller. There are two Ethernet options, i.e., a 4×1 Gb base offering and a 2×10 Gb offering. The I/O hub controller chip provides a pass-through GX I/O bus that is used to drive a GX slot for remote I/O expansion in the 750 model. A second GX I/O bus provides a peak bandwidth of 20 GB/s and drives a second GX slot for remote I/O drawer expansion in the POWER* 750 model or InfiniBand** cluster connect in both the 750 and 755 models.

A single 3-Gb SAS disk controller is integrated onto the I/O backplane and is used to control the eight hot-pluggable 2.5-inch disk drives, as well as the DVD media bay and the tape bay. An optional internal cable feature allows the eight drive bays to be split into two domains whereby the integrated SAS disk controller drives four drive bays and a PCI adapter drives the remaining four drive bays. This provides redundancy in the hard disk drive subsystem or provides a second local boot domain for a second logical partition. Finally, this system supports a redundant array of independent disks (RAID) in numbers 0, 1, 5, 6, and 10 of the internal eight drive bays via an optional pluggable feature that provides the second hard disk controller and dual write cache capability.

Two-socket system architecture

There are four two-socket system offerings, i.e., the 710, 720, 730, and 740 systems. The 710 and 730 systems are packaged in 2U 19-inch rack enclosures and are essentially the same design with the only difference being the number of CPUs installed. The 710 system has one CPU module populated, and the 730 system has both CPU modules populated. This same differentiation exists for the 720 and 740 systems. They are both packaged in 4U 19-inch rack enclosures or a standalone tower enclosure, with the only difference being the number of CPUs installed. The 720 system has one CPU module populated, and the 740 system has both CPU modules populated.

The architecture for these systems is based on a single planar for all four systems, which contains all of the base common function of these systems. This planar contains the

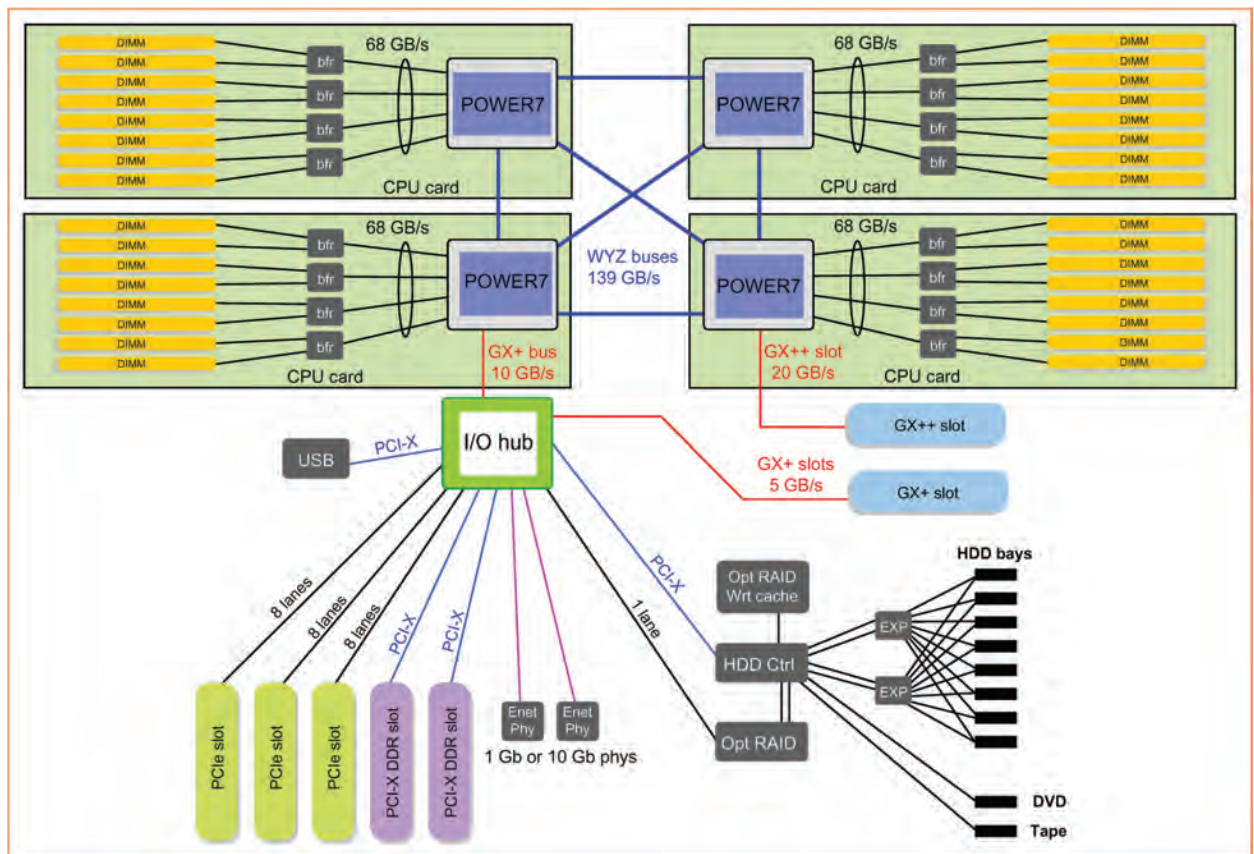


Figure 3

IBM POWER7 four-socket system architecture.

two CPU modules and the base I/O of the system. The rest of the system logic is contained on cards that plug into the base planar, which are optimized for the enclosure in which the system is packaged. One of these enclosure-specific subsystems is the memory. These systems support industry-standard DDR3 memory DIMMs that plug onto a riser card. There are up to four riser cards supported in these systems, and for the larger enclosures of the 720 and 740 systems, the riser card contains eight DDR3 memory DIMM slots, providing a total of 32 DIMM slots. This is four times the number of DIMM slots that were available on the predecessor POWER6 two-socket 4U system. For the 710 and 730 systems that are packaged in the smaller 2U enclosure, the memory riser contains four DDR3 memory DIMM slots, for a system total of 16 DIMM slots (four riser cards times four DIMM slots/riser). Each riser card contains two IBM memory buffer chips and the associated dc-to-dc regulation for the memory DIMMs and the buffer chips.

The I/O subsystem contained on the common planar includes a single IBM I/O hub controller chip that drives the integrated I/O devices, as well as the PCI slots. The

integrated devices include a SAS disk controller and a USB controller. The I/O hub controller drives four PCIe slots. In the 2U enclosure, these PCIe slots support PCIe adapters that conform to the low-profile PCIe form factor [11]. In the 4U rack drawer and tower enclosure, there is additional height that allows full-height PCIe adapters to be installed. The extra space and height of the 4U rack enclosure and tower allow an additional four PCIe slots by installing an I/O riser card into one of the GX slots. This riser contains a second IBM hub controller chip that drives another four PCIe slots. These slots sit above the power supplies, and the adapters that plug into these slots must conform to the low-profile form factor. There are two GX slots on the base I/O backplane that are used for I/O expansion such as the one described above for the 4U enclosure or can be used to support remote I/O expansion to external drawers. Finally, the mechanical packaging drives differences in the disk and media backplanes supported in each platform.

Figure 4 illustrates the system architecture for these two-socket systems. The two processor sockets are interconnected by three nodal fabric buses (W, Y, and Z),

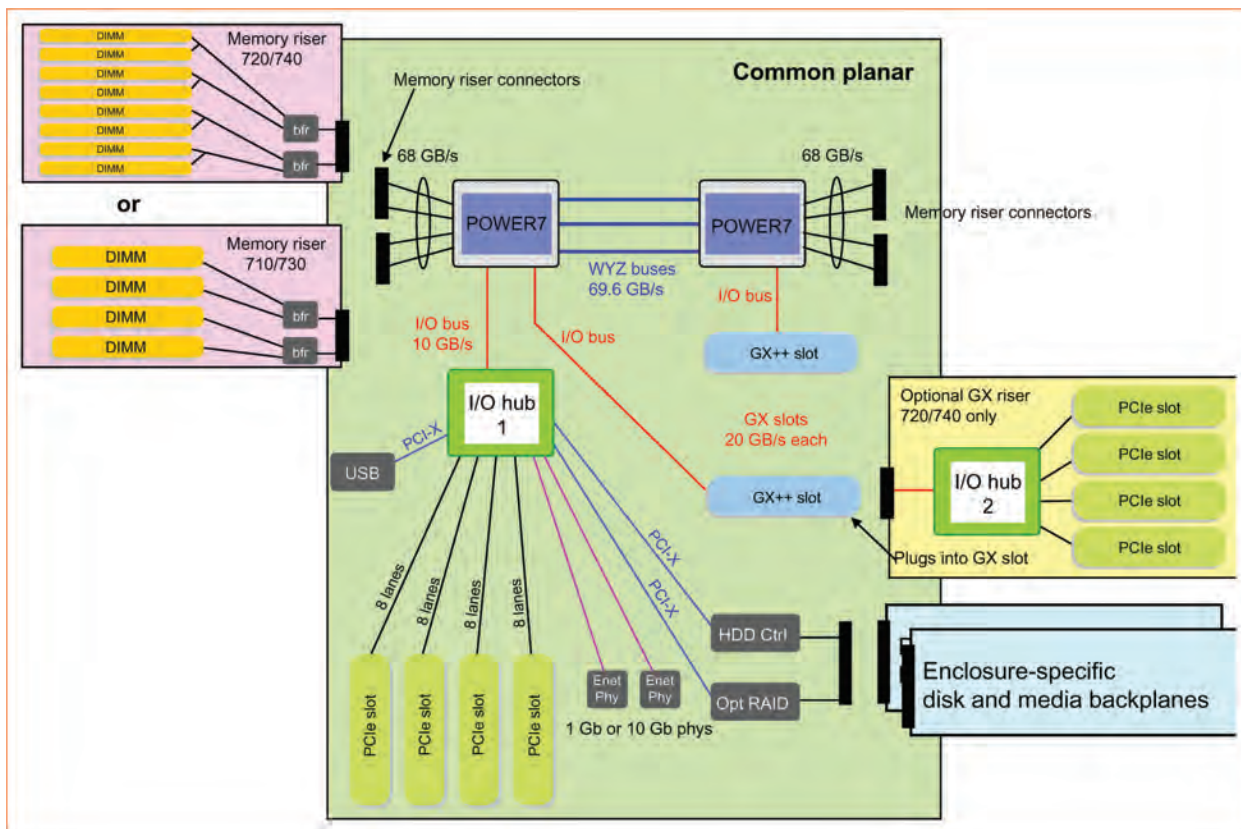


Figure 4
IBM POWER7 two-socket system architecture.

each of which is 4 bytes wide running at 2.9 Gb/s. Each processor module is connected to two memory riser cards and one GX slot. As described earlier, there are two types of memory risers that contain DDR3 DIMM slots operating at 1,066 MHz and connect to the processor module with two memory channels running at a transfer rate of 6.4 Gb/s. The 720/740 system provides a maximum of 32 GB of memory per core for the four-core processor modules or 16 GB of memory per core for the eight-core processor modules. The 710/730 system provides half the memory capacity.

The I/O subsystem is connected to the processors through three GX I/O buses. In the cases where only a single CPU module is populated (710 and 720), one of the GX slots is not functional. One GX I/O bus is connected to the I/O hub controller chip. The second and third GX I/O buses are routed to the GX slots for I/O expansion feature cards. A single 3-Gb SAS disk controller is integrated onto the base planar to support the various disk and media backplanes, and an optional second SAS disk controller is also provided for the RAID operation with a redundant write cache.

Eight-socket systems

There are two highly derivative eight-socket system offerings, the 770 and 780 systems, which are designed to be installed in 19-inch racks. These systems have the same architecture and are differentiated by the processor frequencies supported by each system. The architecture for these systems continues the same building block structure implemented in POWER6 whereby multiple 4U drawers are concatenated together to build larger SMP systems. The minimum configuration is a single drawer, and the maximum configuration is four drawers, which provides for a 64-core maximum system if all sockets contain the eight-core processor modules.

Although the dimensions are the same as those of the predecessor POWER6 eight-socket system, the POWER7 system has implemented several enhancements that are necessary for a balanced system design. A TPMD microprocessor has been implemented in every drawer to support EnergyScale and manage the additional power and cooling of the processor, the memory, and the I/O in the 770/780 systems. The system now supports 250-W processor

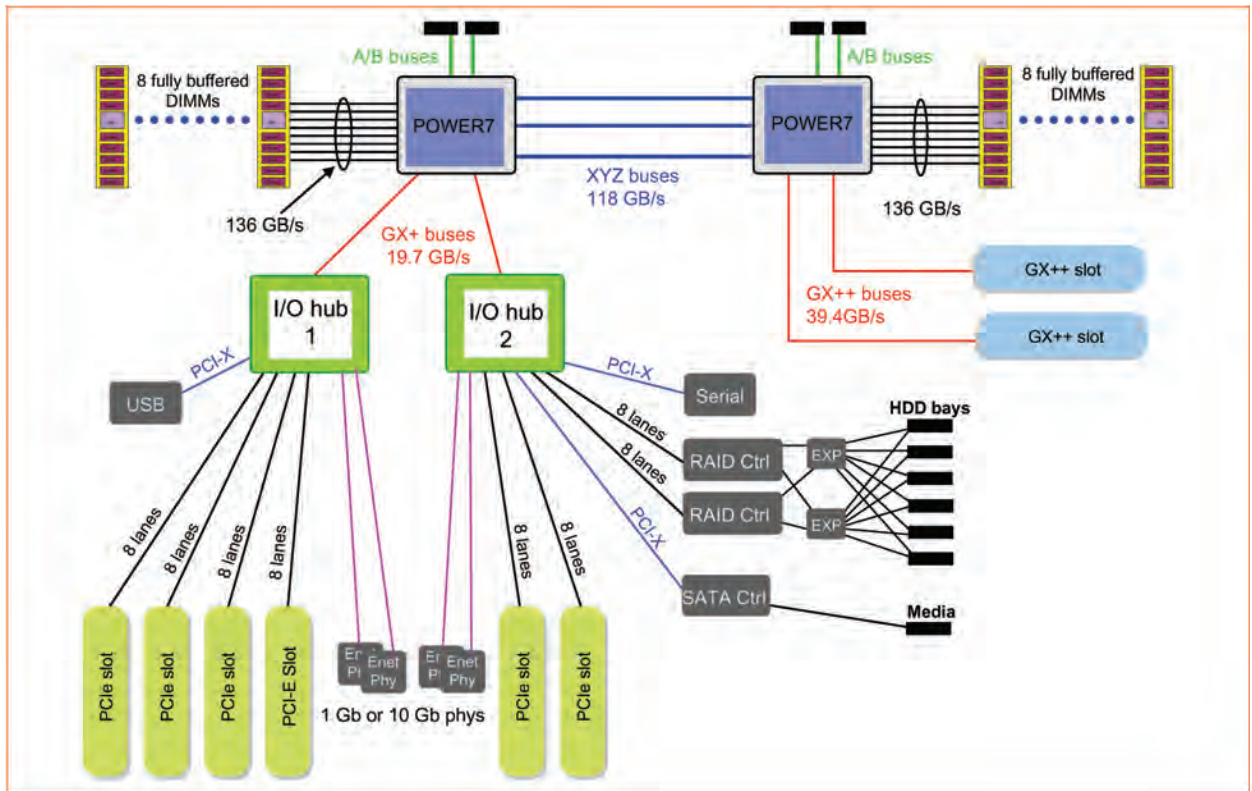


Figure 5

IBM POWER7 eight-socket system drawer block diagram.

sockets with the “overclocking” support of up to 300 W per socket. The total memory capacity has been more than doubled, as compared with POWER6, going from 768 to 2,048 GB of memory. Enhancements to the I/O subsystem include doubling the I/O bandwidth, as well as providing one additional I/O slot and two complete RAID disk controllers integrated on the backplane.

Two primary factors enabled these enhancements. One is the development and the implementation of the IBM custom fully buffered DDR3 memory DIMM to provide the memory density and capacity required for the 770 and 780 systems. As described in a previous section, this DIMM contains one IBM memory buffer chip and up to eight ranks of DDR3 SDRAMs. The SDRAMs used for these DIMMs are of 2-Gb density, as compared with the 1-Gb density used in the DIMMs for the POWER6 eight-socket system. Second, the integration of up to eight ranks of SDRAMs onto a single custom DIMM that is 98 mm tall is 20% more efficient than using the equivalent of four industry-standard 30-mm-tall DIMMs.

As the amount of memory increases in each system, a corresponding increase in the power and the thermal load

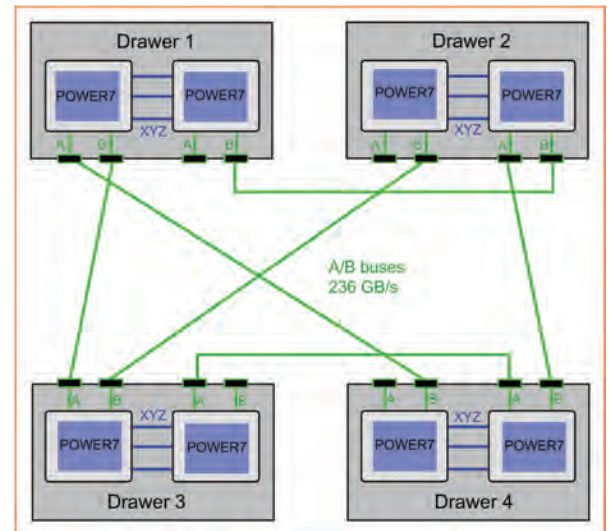


Figure 6

IBM POWER7 eight-socket system SMP connections between drawers.

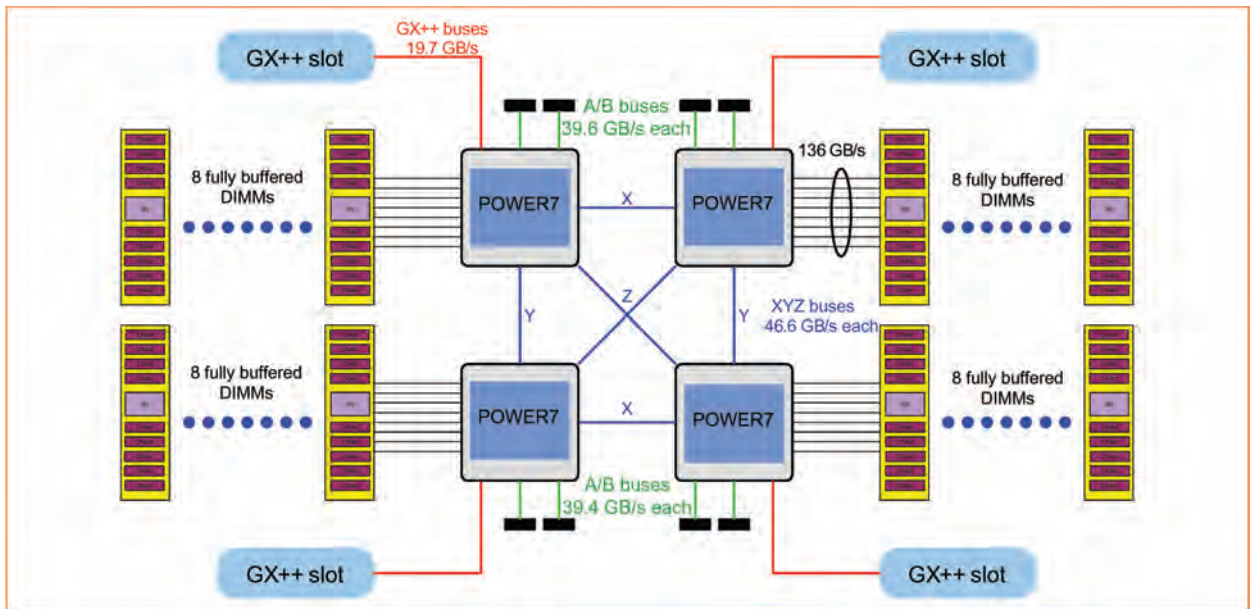


Figure 7
IBM POWER7 32-socket system CEC node.

must be managed. For these systems, the memory is now the largest single user of the bulk power in the system. To mitigate the increase in power, the IBM fully buffered DIMMs use $\times 8$ SDRAMs instead of $\times 4$ SDRAMs. Compared with DIMMs using $\times 4$ SDRAMs, DIMMs with $\times 8$ SDRAMs save significant power since only half the SDRAMs are active on a read operation. Depending on the memory configuration and the workload, the total memory power savings can be at least 30% at the system level [12]. An additional 10% power savings is realized through the use of lower voltage SDRAMs operating at 1.35 V. This compares to the 1.5-V operating voltage of the SDRAMs that are used in most current industry-standard DIMMs.

The second system design enablement was the implementation of the dc-to-dc regulators that convert the 12-V output from the power supply to the lower voltages required by the CPU and the memory. Since the 770 and 780 systems are primarily used as consolidation servers, the failure rates for dc-to-dc regulators drive a requirement for the redundancy of these devices to provide higher system availability. In POWER6, there are three dc-to-dc regulator books that are $N + 1$ redundant, and each regulator book consists of five multiphase buck converters to provide the lower voltages required by the processors and the memory. In addition, these regulator books are concurrently repairable. For POWER7, a much more efficient implementation was implemented that reduced cost and space by moving the redundancy to within the multiphase buck converter

instead of implementing redundant converters. Instead of designing an N phase multiphase buck regulator, the 770/780 systems have implemented multiphase buck regulators that contain $N + 2$ phases. This added redundancy significantly reduces the failure rate of the multiphase buck regulators and eliminates the need for concurrent repair.

Figure 5 shows the architecture of a single drawer for the 770 and 780 systems. Each drawer contains two processor modules and 16 fully buffered DDR3 DIMMs. Each DIMM contains the IBM DDR3 memory buffer chip and DDR3 DRAMs running at 1,066 MHz. Each DIMM is attached to the processor module through a high-speed memory channel running at a transfer rate of 6.4 Gb/s. This system provides a maximum of 32 GB per core for the eight-core processor modules.

The I/O subsystem within a drawer is connected to the processors through four GX I/O buses. Two GX I/O buses provide a 19.7-GB/s peak bandwidth for the integrated I/O and six PCIe $8 \times$ Gen 1 slots. Another pair of GX I/O buses provides a peak bandwidth of 39.4 GB/s and drives two GX slots for remote I/O drawer expansion. **Figure 6** illustrates the drawer-to-drawer SMP connections between the drawers.

Thirty-two-socket system

The largest POWER7 system is the high-end 795 32-socket system. It consists of eight central execution complex (CEC) nodes packaged in a 20U tall cage that fits in a rack that

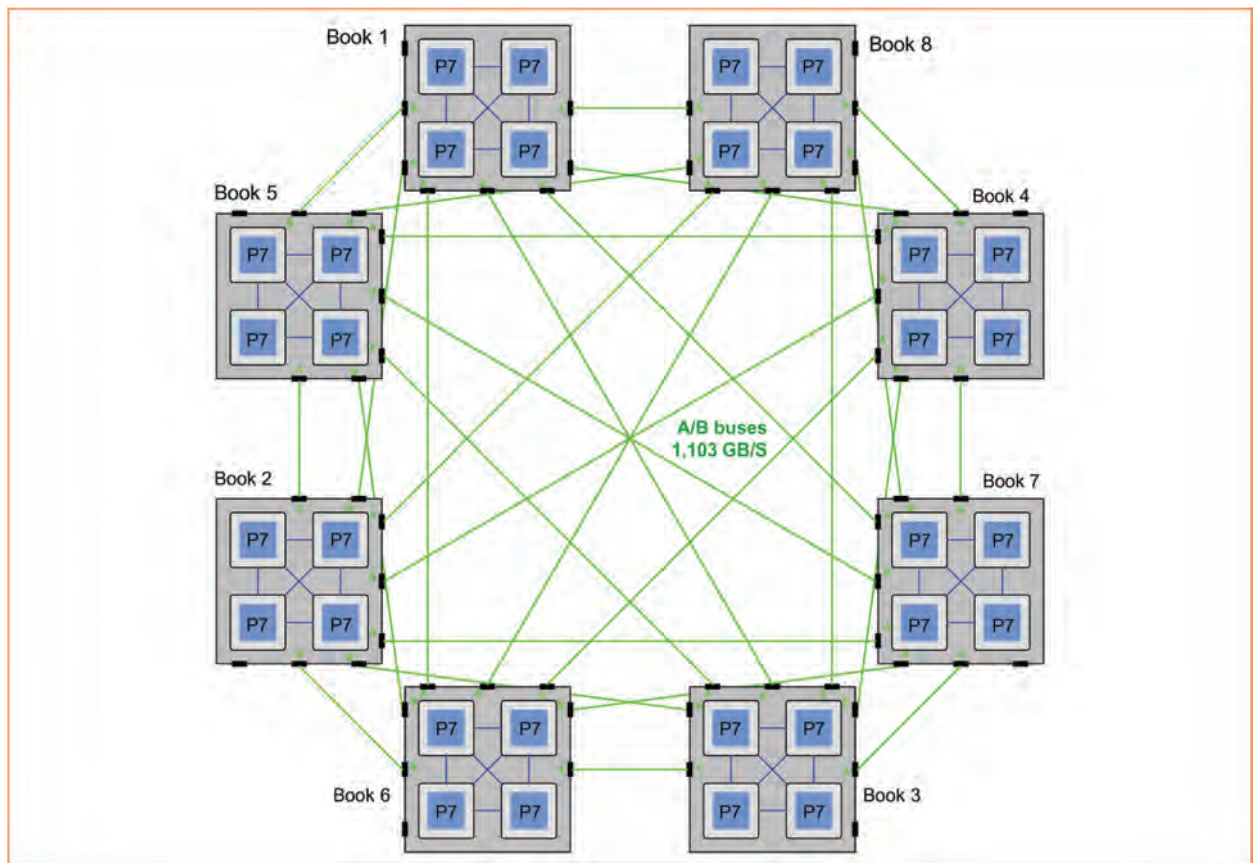


Figure 8

IBM POWER7 32-socket system SMP connections.

is 50 inches deep by 24 inches wide. The 795 system is housed in a 24-inch rack and has the same basic structure as the POWER6 high-end system, with the difference being mostly confined to the CEC nodes that contain the processor and the memory. The CEC node was modified to support four POWER7 sockets using the full-function SCM, and it contains 32 IBM fully buffered DIMMs, i.e., eight per socket.

One of the key innovations for the 795 system is the more efficient delivery of power within the CEC node to mitigate the increase in the power required for the POWER7 CPU “overclocking,” as well as the memory subsystem. A bulk power assembly that is positioned at the top of the 24-inch rack provides a 350-V dc output for each of the CEC nodes. Within the CEC node, a pair of redundant dc-to-dc converter assemblies (DCAs) converts the 350-V dc to approximately 44-V dc that is then distributed within the CEC node to voltage threshold monitors located close to the CPU and the memory. By distributing 44 V to the CEC node instead of 1.25 V that the POWER6 DCAs

distributed, significantly less current is required, which results in a savings of 100 W in distribution losses within the node.

Figure 7 illustrates the architecture of a single CEC node, which contains four processor modules, 32 fully buffered DDR3 DIMMs, and four GX slots. Each DIMM is attached to the processor module through a high-speed memory channel running at a transfer rate of 6.4 Gb/s. This system provides a maximum of 64 GB of memory per core in the four-core TurboCore mode or 32 GB per core for the standard eight-core mode. The system I/O is housed in separate I/O drawers that are connected to the processors through four GX I/O buses, providing a peak book I/O bandwidth of 80 GB/s.

One of the major advantages of the POWER7 processor is the integration of the processor-to-processor switch circuitry for up to 32 sockets or 256 cores. As **Figure 8** shows, the 32 sockets in the 795 server are interconnected without the need for any external logic. This provides several advantages when compared with alternative solutions that

use external switch circuitry such as the Hewlett-Packard Superdome** 2 server [13].

In the Superdome 2 server, 64 sockets (256 cores) of Intel Itanium** processors are interconnected [13]. Each blade requires two agent chips in order to provide a communication path to the crossbar fabric. The crossbar fabric is the coherency vehicle among all of the blades in the system, and it is implemented using 16 crossbar switch chips. Therefore, 80 chips (64 agent chips plus 16 crossbar switches) are required to implement a maximum configured 256-core system. This external crossbar switch implementation not only drives significant complexity and additional cost into the system, but it also adds significant latency for access to the memory that is not local. Some of this latency is reduced by the implementation of a level-4 (L4) cache to store memory lines that have been fetched from the memory on remote blades, but this adds even more cost and complexity to the system.

Conclusion

In this paper, we have described how the latest generation of the IBM POWER servers has fully exploited the powerful POWER7 processor technology to provide a wide range of energy-efficient balanced systems. From the high-end 32-socket 256-core enterprise server to the single-socket four-core blade server, each system has been uniquely designed to achieve the right balance of performance, reliability, and energy efficiency. This wide range of price/performance system offerings has been designed to meet the most demanding environments and workloads across a broad range of business requirements.

*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 The Open Group, Renesas, InfiniBand Trade Association, Hewlett-Packard, or Intel Corporation in the United States, other countries, or both.

References

1. H. B. Bakoglu, G. F. Grohoski, and R. K. Montoye, "The IBM RISC system/6000 processor: Hardware overview," *IBM J. Res. & Dev.*, vol. 34, no. 1, pp. 12–22, Jan. 1990.
2. H. Le, W. Starke, J. Fields, F. O'Connell, D. Nguyen, B. Ronchetti, W. Sauer, E. Schwarz, and M. Vaden, "IBM POWER6 microarchitecture," *IBM J. Res. & Dev.*, vol. 51, no. 6, pp. 639–662, Nov. 2007.
3. B. Sinharoy, R. Kalla, W. J. Starke, H. Q. Le, R. Cagnoni, J. A. Van Norstrand, B. J. Ronchetti, J. Stuecheli, J. Leenstra, G. L. Guthrie, D. Q. Nguyen, B. Blaner, C. F. Marino, E. Retter, and P. Williams, "IBM POWER7 multicore server processor," *IBM J. Res. & Dev.*, vol. 55, no. 3, pp. 1:1–1:29, 2011, Paper 1.
4. M. Broyles, C. Francois, A. Geissler, M. Hollinger, T. Rosedahl, G. Silva, J. Van Heuklon, and B. Veale, *IBM EnergyScale for POWER7 Processor Based Systems*. [Online]. Available: <http://www.ibm.com/systems/power/hardware/whitepapers/energyscale7.html>
5. *UM10204 I²C Bus Specification and User Manual*. Rev. 03, NXP Semicond., Santa Clara, CA, Jun. 2007. [Online]. Available:

- <http://ics.nxp.com/support/documents/interface/pdf/i2c.bus.specification.pdf>
6. B. Sinharoy, R. N. Kalla, J. M. Tendler, R. J. Eickemeyer, and J. B. Joyner, "POWER5 system microarchitecture," *IBM J. Res. & Dev.*, vol. 49, no. 4/5, pp. 505–521, Jul. 2005.
 7. J. M. Tendler, J. S. Dodson, J. S. Fields, Jr., H. Le, and B. Sinharoy, "POWER4 system microarchitecture," *IBM J. Res. & Dev.*, vol. 46, no. 1, pp. 5–25, Jan. 2002.
 8. *PCI Express Base Specification. Rev. 1.1*, PCI-SIG, Beaverton, OR, Mar. 28, 2005. [Online]. Available: [http://wakaba.ojab.ru/wakaba/src/PCI Express Base 1.1 .pdf](http://wakaba.ojab.ru/wakaba/src/PCI%20Express%20Base%201.1.pdf)
 9. *PCI-X Protocol Addendum to the PCI Local Bus Specification. Rev. 2.0a*, PCI-SIG, Beaverton, OR, Jul. 22, 2003. [Online]. Available: http://www.pcisig.com/specifications/pcix/20/ecn_shpc_pcix20.pdf
 10. *Standard for Information Technology Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks Part 3: Carrier Sense Multiple Access With Collision Detection Access Method and Physical Layer Specifications*, IEEE 802.3-2008, Dec. 28, 2008. [Online]. Available: [http://www.techstreet.com/cgi-bin/detail?docno=IEEE%7C802 3 2008&product id=1613165](http://www.techstreet.com/cgi-bin/detail?docno=IEEE%7C802%203%202008&productid=1613165)
 11. *PCI Express Card Electromechanical Specification. Rev. 1.1*, PCI-SIG, Beaverton, OR, Mar. 28, 2005. [Online]. Available: [http://www.pcisig.com/developers/main/training/materials/get_document?doc id=ef7ff351873b6dcbd5e019342e497e4cdd2f0152](http://www.pcisig.com/developers/main/training/materials/get_document?docid=ef7ff351873b6dcbd5e019342e497e4cdd2f0152)
 12. H. Zheng and Z. Zhu, "Power and performance trade-offs in contemporary DRAM system designs for multicore processors," *IEEE Trans. Comput.*, vol. 59, no. 8, pp. 1033–1046, Aug. 2010.
 13. Hewlett-Packard Development Company, *HP Superdome 2: The Ultimate Mission-Critical Platform*, Created Jun., 2010, White Paper 4AA1-7762ENW. [Online]. Available: <http://h18006.www1.hp.com/storage/pdfs/4AA1-7762ENW.pdf>

Received November 29, 2010; accepted for publication February 21, 2011

Ronald X. (Ron) Arroyo *IBM Systems and Technology Group, Austin, TX 78758 USA (arroyor@us.ibm.com)*. Mr. Arroyo joined IBM in 1981 after receiving a B.S. degree in electrical engineering from the University of Texas. He is a Distinguished Engineer in the POWER Systems Design and Architecture department. He has worked on the development of several POWER systems designs beginning with POWER3* and led the development of IBM's Model 570 and 770/780 midrange servers. Mr. Arroyo currently has overall responsibility for IBM's POWER systems design and architecture.

Raymond J. Harrington *IBM Systems and Technology Group, Poughkeepsie, NY 12601 USA (rharrington@us.ibm.com)*. Mr. Harrington received a B.S.E.E. degree from Clarkson University, Potsdam, NY and an M.S.E.E. degree from Syracuse University, Syracuse, NY. He has designed various elements of System p* and System z* servers including elements of the service systems, power systems, and clock distribution. He has been the lead architect for the IBM POWER 595 and POWER 795 high-end servers. Mr. Harrington holds 17 patents.

Steven P. Hartman *IBM Server Group, Austin, TX 78758 USA (sph@us.ibm.com)*. Mr. Hartman received his B.S.E.E. degree from Penn State University, University Park, in 1974. He joined IBM in 1978 and worked in the design of Optical Character Recognition Systems and in system architectures for banking systems. He moved to Austin in 1991 and served as the lead architect and technical lead for a variety of systems from RS/6000* workstations to high-end POWER4 servers and high-performance computers. Currently he is the chief architect for POWER blades.

Thoi Nguyen *IBM Systems and Technology Group, Austin, TX 78758 USA (thoi@us.ibm.com).* Mr. Nguyen received a B.S.E.E. and M.S.E.E. degrees from Tennessee Technological University, Cookeville and joined IBM Lexington, Kentucky, in 1977. He is a Senior Technical Staff Member and the Chief Architect of POWER7 755/750, 740/720, and 730/710 systems. Mr. Nguyen worked in development of printer, high-speed optical fiber interconnect, set-top box design and servers. He architected and designed POWER4, POWER5, and POWER6 low-end servers. Mr. Nguyen achieved an IBM Fifth Invention Plateau and holds 16 patents.

PERCS: The IBM POWER7-IH high-performance computing system

R. Rajamony
L. B. Arimilli
K. Gildea

In 2001, the Defense Advanced Research Projects Agency called for the creation of commercially viable computing systems that would not only perform at very high levels but also be highly productive. The forthcoming POWER7®-IH system known as Productive, Easy-to-use, Reliable Computing System (PERCS) was IBM's response to this challenge. Compared with state-of-the-art high-performance computing systems in existence today, PERCS has very high performance and productivity goals and achieves them through tight integration of computing, networking, storage, and software. This paper describes the PERCS hardware and software, along with many of the design decisions that went into its creation.

Introduction

Although silicon technologies (e.g., multicore dies and smaller device sizes) continue to improve generation after generation [1], surrounding technologies in high-performance computing (HPC) systems such as interconnect bandwidth, memory densities and bandwidths, power packaging and cooling, and storage densities and bandwidths do not accordingly scale. For instance, while the per-processor Linpack performance [2] shows a steady improvement over time, interconnect-intensive metrics such as G-RandomAccess and G-FFTE [3], at best, show a modest improvement on a per-processor level.

The Productive, Easy-to-use, Reliable Computing System (PERCS) program was initiated in 2001 as IBM's response to the Defense Advanced Research Projects Agency (DARPA) High-Productivity Computing Systems (HPCSs) program [4]. A significant point of emphasis in the HPCS program was productivity: "These systems will improve by orders of magnitude the effectiveness of humans and computers in solving the problems in the high-performance computing domain" [5].

The challenge of building a highly productive HPCS forced us to recognize early on that an integrated approach to system construction was essential. Accordingly, varied aspects of an HPC system such as packaging, power delivery, cooling, architecture, topology, interconnect,

communications software, compilers, operating system, and even programming language are tightly integrated in PERCS.

The system uses POWER7* microprocessors arranged on a quad-chip module that runs a single operating-system image managing 32 homogeneous high-performance compute cores. The system uses a two-level direct-connect interconnect topology that fully connects every element in each of the two levels. Judicious use of the copper conductor transport at small scales and fiber optics at larger scales permits an innovative packaging solution that can provide more than 90 tera-floating-point operations per second of compute capability within a single rack.

The PERCS system has a memory bandwidth to compute the capability [in bytes/second per floating-point operation (FLOP)] in excess of 0.5 and an interconnect bisection bandwidth to compute the capability (in bytes/second per FLOP) in excess of 0.08. These metrics, as well as the homogeneous nature of the compute facilities, enable very high execution time productivity and improvements in programmer workflows pertaining to runtime issues.

The software layers in PERCS tightly integrate with the hardware. A user-level packet constructed by user-level code or the communications software layer can atomically update the contents of the memory in any part of a PERCS system, exercising various hardware components in the system. The programming language X10 [6] was concomitantly developed with the system architecture, and each influenced many aspects of the other, including the notion of "places" in X10 and the PERCS global-address space.

Digital Object Identifier: 10.1147/JRD.2011.2109230

©Copyright 2011 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied by any means or distributed royalty free without further permission by computer-based and other information-service systems. Permission to republish any other portion of this paper must be obtained from the Editor.

0018-8646/11/\$5.00 © 2011 IBM

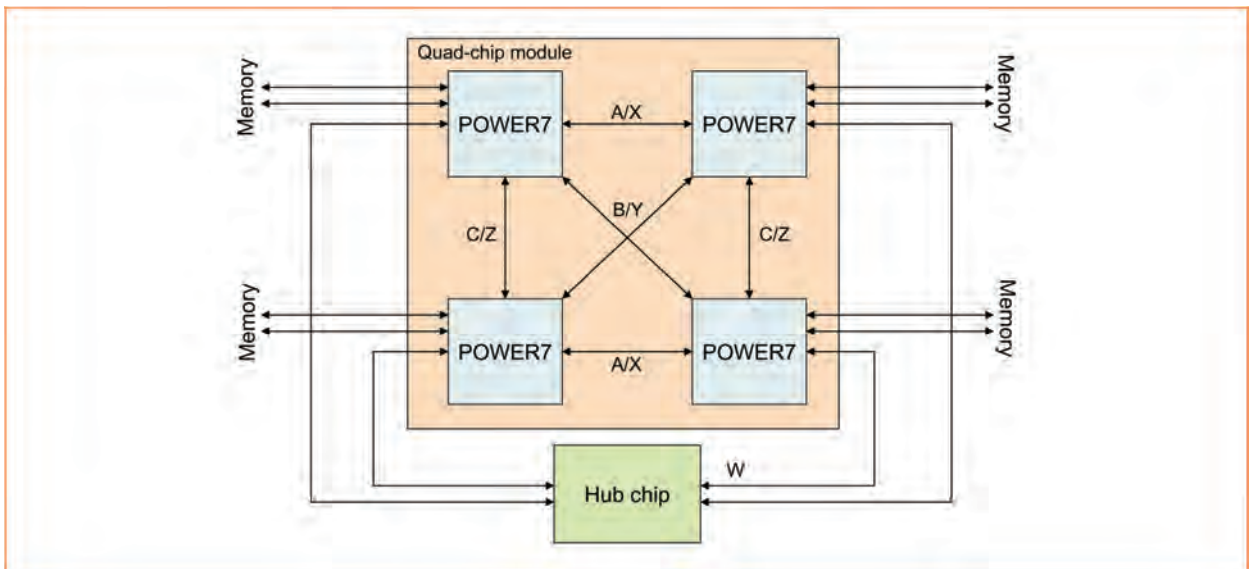


Figure 1
Compute node structure.

The large level of integration in PERCS improves the size and the scale of POWER7-IH systems and dramatically simplifies the machine room. The significant reduction in the number of components also improves the reliability characteristics of the system. Advantages of the PERCS approach include the following:

- *HPC network*—No host channel adapters, no external switches, and half the physical interface and cables of an equivalent fat-tree structure with the same bisection bandwidth.
- *Storage*—No fiber channel adaptors, no fiber channel cables, and no external storage controllers.
- *Server*—No external PCI Express** (PCIe**) channel controllers, no external memory controllers, and no external cache chips.

This paper is organized as follows: We first present the different components of the PERCS hardware. The PERCS system structure and organization are then described, followed by a brief summary of the PERCS performance capabilities. The PERCS two-level direct-connect interconnect topology is then explained. We follow this with a description of the IBM hub chip, which is the gateway to the system interconnect. The routing operations performed by the hub chip are then elaborated. A petascale system such as PERCS requires significant attention to reliability, availability, and serviceability (RAS), as well as failure management in order to be productive; these features are then described. We conclude this paper with a brief description of the PERCS software stack.

System components

Figure 1 shows the abstract structure of a compute node in a PERCS system. There are four POWER7 chips in a node with a single operating-system image that controls resource allocation. Applications executing on the compute node can utilize 32 cores, 128 simultaneous multithreading (SMT) threads, 8 memory controllers, up to 512 GB of memory capacity, more than 900 giga-floating-point operations per second of compute capacity, and more than 500 GB/s of memory bandwidth. The four POWER7 chips are cache coherent and are tightly coupled using three pairs of buses.

The IBM hub chip in **Figure 1** completes the compute node, providing network connectivity to the four POWER7 chips. The hub chip participates in the cache coherence protocol within the node and serves not only as an interconnect gateway to the four POWER7 chips that connect to it but also as a switch that routes traffic between other hub chips. Therefore, a PERCS system requires no external network switches or routers with considerable savings in the switching components, cabling, and power.

POWER7 chip

Figure 2 shows a logical view of the POWER7 chip that constitutes the compute heart of the POWER7-IH system. It incorporates eight superscalar speculative multithreaded cores, a large cache subsystem, two memory controllers with eight memory channels (not shown in the figure), and several high-bandwidth links to connect to other parts of the PERCS subsystem.

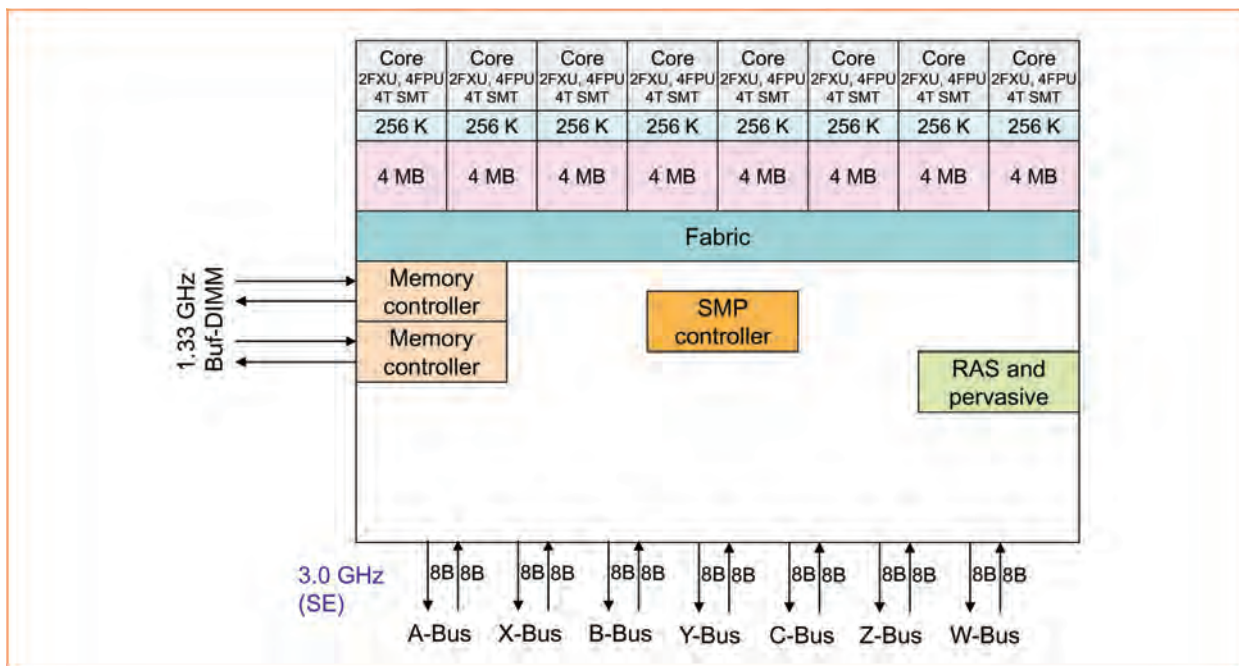


Figure 2

POWER7 chip structure.

Through its eight cores, each POWER7 chip is capable of performing 64 FLOPs or 32 integer operations per cycle. Further information on the compute capabilities of POWER7, including its performance capabilities, is available elsewhere [7, 8].

Memory subsystem

The two memory controllers in each POWER7 chip connect through eight memory channels to multiported dual inline memory modules (DIMMs) that carry a buffer chip called the SuperNova. Each POWER7 chip can currently accommodate up to 128 GB of memory capacity, with additional memory capacity achievable through denser dynamic random-access memory chips. The memory controllers within POWER7 and the SuperNova chips have been designed to achieve high performance not only for regular access patterns but also for the random-access pattern characteristic of many sparse matrix and graph computation problems. The large reorder buffer within the memory controller, as well as separate 64-entry READ and WRITE queues within each controller, permits a significant tolerance for accesses that target DIMM banks in bursts. The memory subsystem can also perform 64-byte partial cache loads and stores, which improves the efficiency and the bandwidth for applications with a poor spatial locality.

Each memory controller also has the ability to atomically perform READ-MODIFY-WRITE operations on the data resident

within the DIMMs that it controls. This feature of the PERCS system is described in greater detail when we discuss the system interconnect. Further information on the memory subsystem of POWER7 systems can be found elsewhere [7, 9].

Storage

There are no separate storage controller elements in a POWER7-IH system. A group of POWER7 chips that are identical with each other with respect to compute resources in the POWER7-IH system are typically allocated to execute the General Parallel File System (GPFS) [10] network shared disk (NSD) server on behalf of the other compute resources in the system. Roughly 1.5% of the raw compute resources of a PERCS system are set aside for storage management purposes. The raw storage devices are also integrated into POWER7-IH racks, providing for a modular building block that is self-sufficient in the compute, the interconnect, and the storage. These devices are connected to the compute node through Serial-Attached Small Computer System Interface [Serial-Attached SCSI (SAS)] adaptor cards that, in turn, connect to the hub chip through a PCIe 2.1 interface [11].

Hub chip

Figure 3 shows an overview of the hub chip. The main purpose of the IBM hub chip is to interconnect tens of thousands of compute nodes and to provide input/output

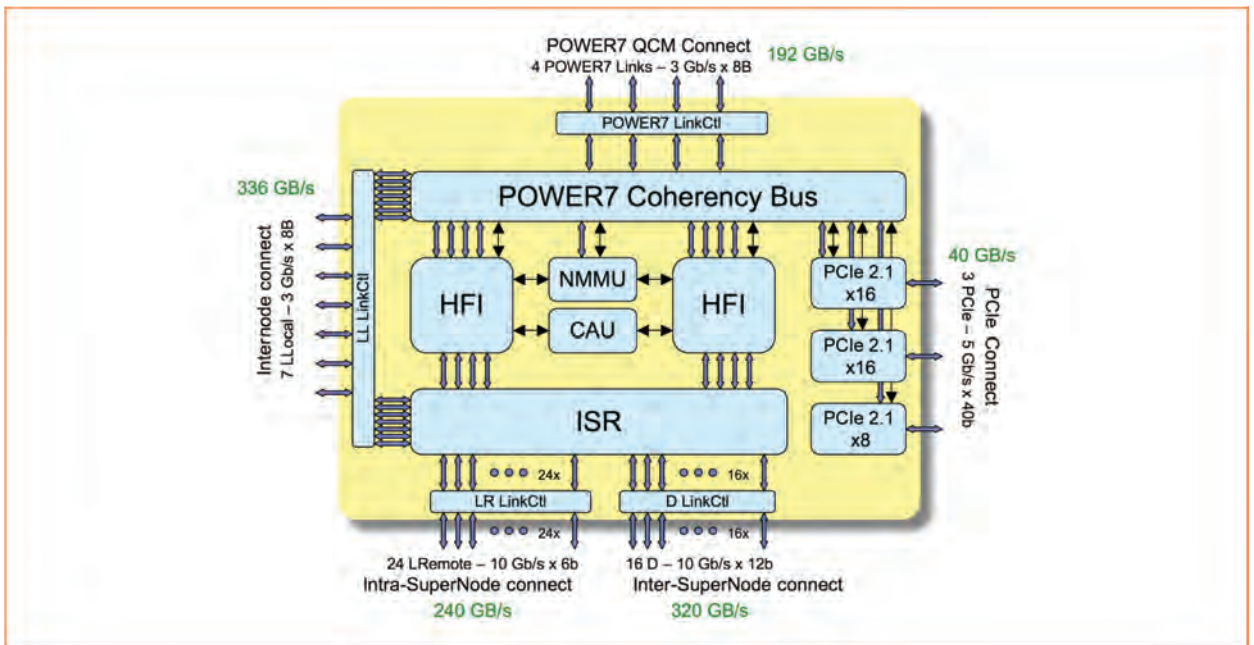


Figure 3
IBM hub-chip overview.

(I/O) services. The hub design provides ultralow latencies at high bandwidth, dramatically improving the scalability of applications written using various application programming interfaces (APIs) such as message passing interface (MPI), sockets, and partitioned global-address space (PGAS) languages [6].

The hub chip also improves the performance and the cost of an HPC storage subsystem by requiring no fiber channel host bus adapters, no external switches, no storage controllers, and no direct-attached storage within the compute nodes. The hub chip also obviates the need for external PCIe controllers by integrating them on the chip.

Key functions used by the software are accelerated in the hardware by the hub chip. The collective acceleration unit in the hub chip speeds up collective (including synchronization) operations that are often a significant scalability impediment to HPC applications. The hub chip also employs a memory management unit that is kept consistent with the translation lookaside buffers (TLBs) on the compute cores. This enables an application running on one compute node to use program-level effective addresses to operate on data located on another compute node. Finally, the hub chip also has special facilities to enable certain operations to be atomically performed in the compute node's memory without involving any of the compute node's cores.

The hub chip is implemented using 45-nm lithography Cu silicon-on-insulator technologies. The chip is 582 mm² in size with 440 million transistors and 13 levels of on-chip

metal wiring. There are more than 3,700 signal I/O and more than 11,000 total I/O pins. The hub chip is integrated along with optics modules into a 58-cm² glass-ceramic land grid array module.

System organization and key metrics

The PERCS system is organized as follows. Eight compute nodes (shown in Figure 1) are physically contained within a drawer that is 3.6 inches high (2U), 34.4 inches wide, and 54 inches deep. In the canonical arrangement, four of such drawers are organized into a structure known as a supernode by virtue of how they are connected with each other. Supernodes are placed into 39-inch racks with either two or three supernodes per rack. The two-supernode configuration can contain more storage drives than the three-supernode configuration. **Figure 4** shows the structure of a PERCS rack and drawer.

The maximal PERCS system can contain 512 supernodes. Several other configurations are also possible, including those in which drawers are not organized into supernodes. Raw storage devices are contained in a storage enclosure that has the same width and depth as the compute drawers but twice the height.

The entire system is water cooled through the use of plumbing tubes that run across or over different system components including processor chips, memory, and hub chips. Deionized water is used internally to a rack and a set of four water-conditioning units that transfer heat between the

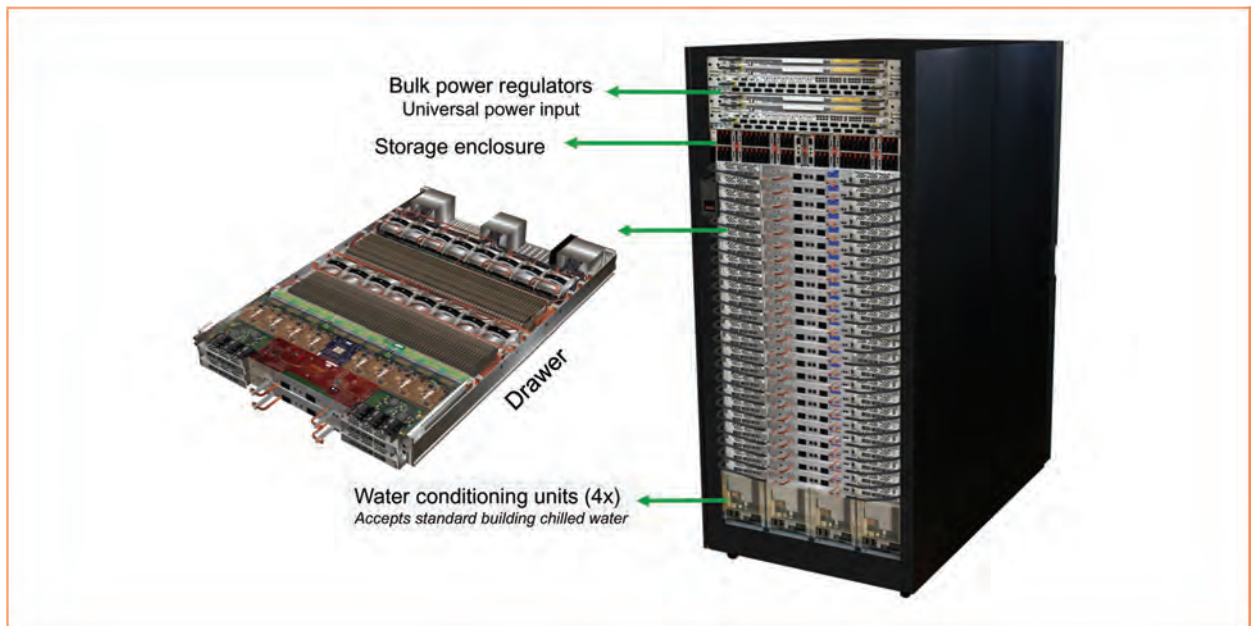


Figure 4

POWER7-IH rack and drawer.

Table 1 Key PERCS metrics.

<i>Metric</i>	<i>Value</i>
Compute performance	0.98 TFLOPs per compute node
Memory bandwidth	0.51 TB/s per compute node
Cores	32 per compute node
Memory capacity	Exceeds 128 GB/compute node
Interconnect bandwidth	192 GB/s per compute node (peak, bi-directional)

rack water and the external (untreated) chilled data-center water.

Certain salient metrics of the PERCS system are outlined in **Table 1**. Srinivas et al. have described metrics related to the POWER7 processor, including the performance [8].

PERCS topology

Two key design goals for PERCS were to dramatically improve the bisection bandwidth over other topologies such as fat-tree interconnects and to eliminate the need for external switches. With these goals in mind, the hub chip was designed to support a large number of links that connect it to other hub chips. These links are classified into two categories “L” and “D” that permit the system to be organized into a two-level direct-connect topology. Figure 3 and **Figure 5** illustrate these concepts [12].

Every hub chip has 31 L links that are used to fully connect 32 hub chips with a supernode into an all-to-all topology. Within this group of 32 hub chips, each chip has a direct communication link to every other chip. The hub-chip implementation further divides the L links into two categories: seven electrical LL links with a combined bandwidth of 336 GB/s and 24 optical LR links with a combined bandwidth of 240 GB/s. The LL links are used to interconnect hub chips within a single drawer. The LR links are used to interconnect hub chips across the four drawers in one supernode.

Every hub chip also has 16 D links that are used to connect to other supernodes with a combined bandwidth of 320 GB/s. The topology maintains at least one D link between every pair of supernodes in the system, although smaller systems can employ multiple D links between supernode pairs.

Since the hub chip is connected to the POWER7 chips in the compute node at a bandwidth of 192 GB/s and has 40 GB/s of bandwidth for the general I/O, the peak switching bandwidth of the hub chip exceeds 1.1 TB/s. The PERCS compute node has a very high ratio of the injection bandwidth to or from the compute POWER7 chips and the network bandwidth to or from the hub chip. When all links are populated and operate at peak bandwidths, the ratio of the injection bandwidth to the network bandwidth is 1.0 : 4.6. Note that by performing the dual roles of switch and interconnect gateway, the majority of the traffic through the hub chip will typically be destined for other compute nodes.

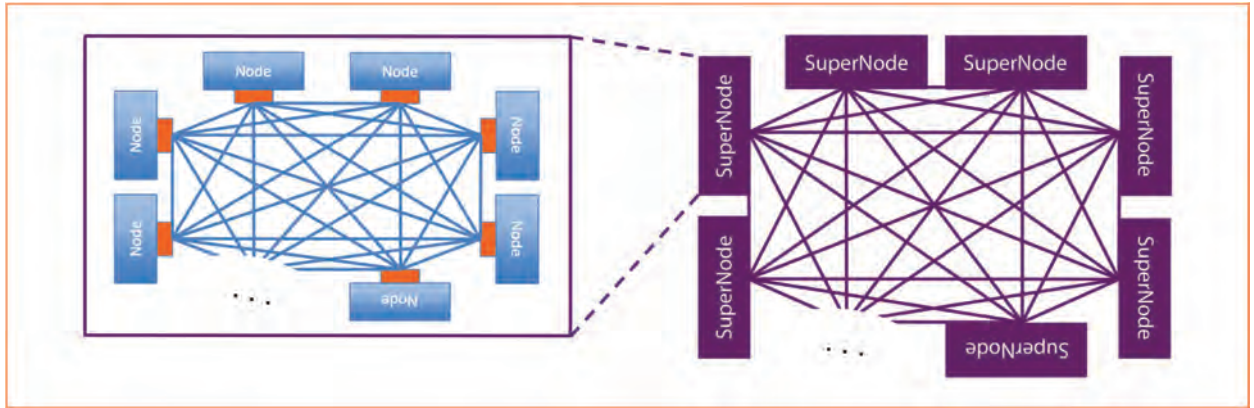


Figure 5

Direct connections between nodes in a supernode and supernodes in the system. LL and LR links connect nodes within a supernode. Supernodes shown on the right-hand side are connected with D links.

The topology used by PERCS permits routes to consist of very small numbers of hops. Within a supernode, any compute node can communicate with any other compute node using a distinct L link. Across supernodes, a compute node has to employ at most one L hop to get to the “right” compute node within its supernode that is connected to the destination supernode (recall that every supernode pair is connected by at least one D link), a D hop to reach that supernode, and at most one L hop at the destination supernode to reach the destination compute node.

Hub chip details

A significant component of our scaling solution is a new switchless interconnect with very high fan-out organized into the two-level direct-connect topology described above. The hub chip implements this interconnect and is described in greater detail in this section. Figure 3 shows the different hub-chip components pictorially.

PowerBus interface

The PowerBus interface enables the hub chip to participate in the coherency operations taking place between the four POWER7 chips in the compute node. The hub chip is a first-class citizen in the coherence protocol and has visibility to coherence transactions taking place in the node, including TLB-related coherence operations.

Host fabric interface

The two host fabric interface (HFI) units in the hub chip manage the communication to and from the PERCS interconnect. The HFI was designed to provide user-level access to applications. The basic construct provided by the HFI to applications for delineating different communication

contexts is the “window.” The HFI supports many hundreds of such windows, each with its associated hardware state.

An application invokes the operating system and, thus, the hypervisor to reserve a window for its use. The reservation procedure maps certain structures of the HFI into the application’s address space with the window control being possible from that point onwards through user-level READS and WRITES to the HFI mapped structures.

The HFI supports three APIs for communication:

- *General packet transfer* can be used for composing unreliable protocols, as well as reliable protocols such as that needed for the MPI through higher levels of the software stack.
- *Global-address space operations and active messaging* can be used by user-level codes to manipulate directly memory locations of a task executing on a different compute node. The nest memory management unit (NMMU) provides support for these operations.
- *Direct Internet Protocol (IP)* transfers.

The HFI can extract data that needs to be communicated over the interconnect either from the POWER7 memory or directly from the POWER7 caches. The choice of the source is transparent, with the data being automatically sourced from the faster location (caches can typically source data faster than the memory). In addition to writing network data to the memory, the HFI can also inject network data directly into a processor’s level 3 (L3) cache, thus lowering the data access latency for the code executing on that processor.

Five primary packet formats are supported, i.e., immediate sends; first-in, first-out (FIFO) send and receive; IP; remote direct memory access (RDMA); and atomic updates.

A new PowerPC* instruction, ICSWX, is used to implement immediate sends [13]. This instruction forces a cache line directly to the HFI for interconnect transmission and is the lowest latency (at the expense of the bandwidth) communication mechanism for sending packets that are less than a cache line in size.

The FIFO send/receive mode permits an application to use a staging area for both sending and receiving data. An application can prereserve a portion of its address space to serve as circular FIFO buffers. After composing packets in the send FIFO, the application “triggers” the HFI by writing an 8-byte value to a per-window trigger location. In this mode, incoming packets are written to the receive FIFO by the HFI and can be then processed by the application. An 8-byte WRITE to another location informs the HFI of the space that it can reuse in the receive FIFO.

The HFI supports two forms of IP transfers. IP packets can be transferred to and from the FIFO buffer (see above). IP packets can be also described with scatter or gather descriptors with the HFI assembling or disassembling data.

A variety of RDMA mechanisms are supported. In addition to traditional memory-to-memory transfers, the HFI also supports transfers between the FIFO buffer and memory. Since these are asynchronous operations, completion notifications permit an application to implement READ and WRITE fences.

A final packet format permits an application to specify atomic updates to remote memory locations. Fixed-point operations such as ADD, AND, OR, XOR, and compare and swap with and without fetch for multiple data sizes (8, 16, 32, and 64 bits) are supported. Sequence numbers are used to ensure proper reliable operation of all atomic updates, with an optimized mode permitting up to four operations to be packed per cache line at a coarser reliability granularity. Collective packets are also supported, and the operation is described in greater detail in the section describing the collective acceleration unit.

Integrated switch router

The integrated switch router (ISR) implements the two-tiered full-graph network described in the section on the PERCS topology. It is organized as a 56×56 full crossbar that operates at up to 3 GHz. In addition to the 47 L and D ports previously described, the ISR also has eight ports to the two local HFIs, as well as one service port.

The ISR uses both input buffering and output buffering with a packet replay mechanism to tolerate transient link errors. This feature is particularly important since the D links can be several tens of meters in length. The ISR operates in units of 128-byte flow control digits (FLITs) with a maximum packet size of 2,048 bytes. Messages are composed of multiple packets with the packets making up a message being potentially delivered out of order.

HPC applications benefit from having access to a single global clock across the entire system. The ISR implements a

global clock feature whereby a clock onboard is globally distributed across the interconnect and kept consistent with the clocks on other hub chips. More details of the ISR as it pertains to routing are described in the section on routing.

Collective acceleration unit

Many HPC applications perform collective operations with the application being able to make forward progress not only after every compute node has completed its contribution to the collective operation but also after the results of the collective operation are disseminated back to every compute node (e.g., barrier synchronization or a global sum). The hub chip provides a specialized hardware to accelerate frequently used collective operations.

A specialized arithmetic-logic unit (ALU) within the collective acceleration unit (CAU) implements multicast, barrier, and reduction operations. For reductions, the ALU supports the following operations and data types:

- *Fixed point*—NOP, SUM, MIN, MAX, OR, AND, and XOR (signed and unsigned).
- *Floating point*—MIN, MAX, SUM, and PROD (single and double precision).

The software organizes the CAUs in the system into collective trees. For a broadcast operation, the arrival of an input on one link causes its forwarding on all other links; for a reduce operation, arrivals on all but one link cause the reduction result to be forwarded on the remaining links. There is one CAU in each hub chip, and a link in the CAU tree could map to a path in the network made up of more than one link. A multiple-entry content-addressable memory structure per CAU supports multiple independent trees that can be concurrently used by different applications, for different collective patterns within the same application, or some combination thereof.

Reliability and pipelining are afforded using sequence numbers and a retransmission protocol. Each tree has exactly one participating HFI window on any involved node. The tree can be set up such that the order in which the reduction operations are evaluated is preserved from one run to another. Programming models such as the MPI [14], which permit programmers to require collective operations to be executed in a particular order, can benefit from this feature.

NMMU

A key facility for high-performance global-address space languages such as Unified Parallel C (UPC) [15], Coarray Fortran [16], and X10 [6] is a low-overhead mechanism for the user-level code to operate on the address space of processes executing on other compute nodes. The NMMU in the hub chip facilitates such operations enabling the user-level code to create a global-address space against

which to perform operations. This facility in PERCS is called “global shared memory.”

A process executing on a compute node can register its address space, permitting interconnect packets to manipulate the registered region directly. Registering a portion of the address space results in the NMMU being able to reference a page table that maps effective addresses to the real memory. A cache of the mappings is also maintained within the hub chip and can map the entire real memory of most installations.

Incoming interconnect packets that reference memory such as RDMA packets and packets that perform atomic operations contain both an effective address and information that pinpoints the context in which to translate the effective address. This greatly facilitates global-address space languages by permitting such packets to contain easy-to-use effective addresses.

I/O connectivity

The hub chip has three PCIe 2.1 ports. Two of the ports are $\times 16$ and support $\times 16$, $\times 8$, $\times 4$, and $\times 1$ connections. The third port is $\times 8$ and supports $\times 8$, $\times 4$, and $\times 1$ connections. The ports are all backward compatible up to generation 1.1a. The hub chip supports the “hot-plug” capability.

PERCS global counter

The PERCS global-counter design provides a counter on each ISR that is synchronized across the entire cluster. This counter is used to address the operating-system jitter, to tune and debug applications and system software, and to support a globally synchronous trigger for sampling HFI and ISR performance counters. Software applications can access the global counter located in their group on each drawer in the cluster. The goal is to have minimum skew between the 16,000 global counters in a maximal PERCS system. The network management software has a role in setting up and maintaining the global counters, but most of the synchronization function resides in the ISR hardware.

Each global counter consists of a set of ISR registers that can be initialized by a service processor scan before runtime and then either read or written during runtime by either the service processor or the hypervisor. Network management software uses the service processor to access the eight ISR global counters located in each drawer. The user-level software can read the global-counter value to perform higher level synchronization functions.

Routing in PERCS

Routes in PERCS can be classified into “direct” and “indirect” routes. A direct route employs the shortest path between any two compute nodes in the system. Since a pair of supernodes can be connected together by more than one D link, there can be multiple shortest paths between a given set of compute nodes. With only two levels in the topology, the longest direct route can have at most three hops

made up of no more than two L hops and at most one D hop. We call this an L–D–L route.

PERCS supports indirect routes to provide increased point-to-point bandwidth and to guard against potential hot spots in the interconnect. An indirect route is the one that has an intermediate compute node in the route that resides on a different supernode from that of the source and destination compute nodes. An indirect route must employ the shortest path from the source compute node to the intermediate one and the shortest path from the intermediate compute node to the destination compute node. The longest indirect route can have at most five hops made up of no more than three L hops and at most two D hops. We call this an L–D–L–D–L route. **Figure 6** illustrates the direct routing and the indirect routing within the PERCS system.

A specific route can be selected in three ways when multiple routes exist between a source–destination pair. First, the software can specify the intermediate supernode but let the hardware determine how to route to and then from the intermediate supernode. Second, the hardware can select among the multiple routes in a round-robin manner for both direct and indirect routes. Finally, the hub chip also provides support for route randomization whereby the hardware can pseudorandomly pick one of the many possible routes between a source–destination pair. Hardware-directed randomized route selection is available only for indirect routes. These routing modes can be specified on a per-packet basis.

The right choice between the use of direct- versus indirect-route modes depends on the communication pattern(s) used by applications. Direct routing will be suitable for communication patterns where each node has to communicate with a large number of other nodes as that with spectral methods. Communication patterns that involve small numbers of compute nodes will benefit from the extra bandwidth offered by the multiple routes with indirect routing.

Routing is accomplished using static route tables placed in the routers (ISR). These route tables are set up during system initialization and are dynamically adjusted as links go down or come up during operation. Packets are injected into the network with a destination identifier and the route mode. The route information is picked up from the route tables along the route path based on this information. Packets injected into the interconnect by the HFI employ source route tables with the route partially determined. Per-port route tables are used to route packets along each hop in the network. Separate route tables are used for intersupernode and intrasupernode routes.

Virtual channels (VCs) are used to prevent deadlocks. We use the position of the current hop within the full route to select which VC to use. Based on the worst case route in the system being L–D–L–D–L, there are three VCs assigned to L links and two VCs assigned to D links.

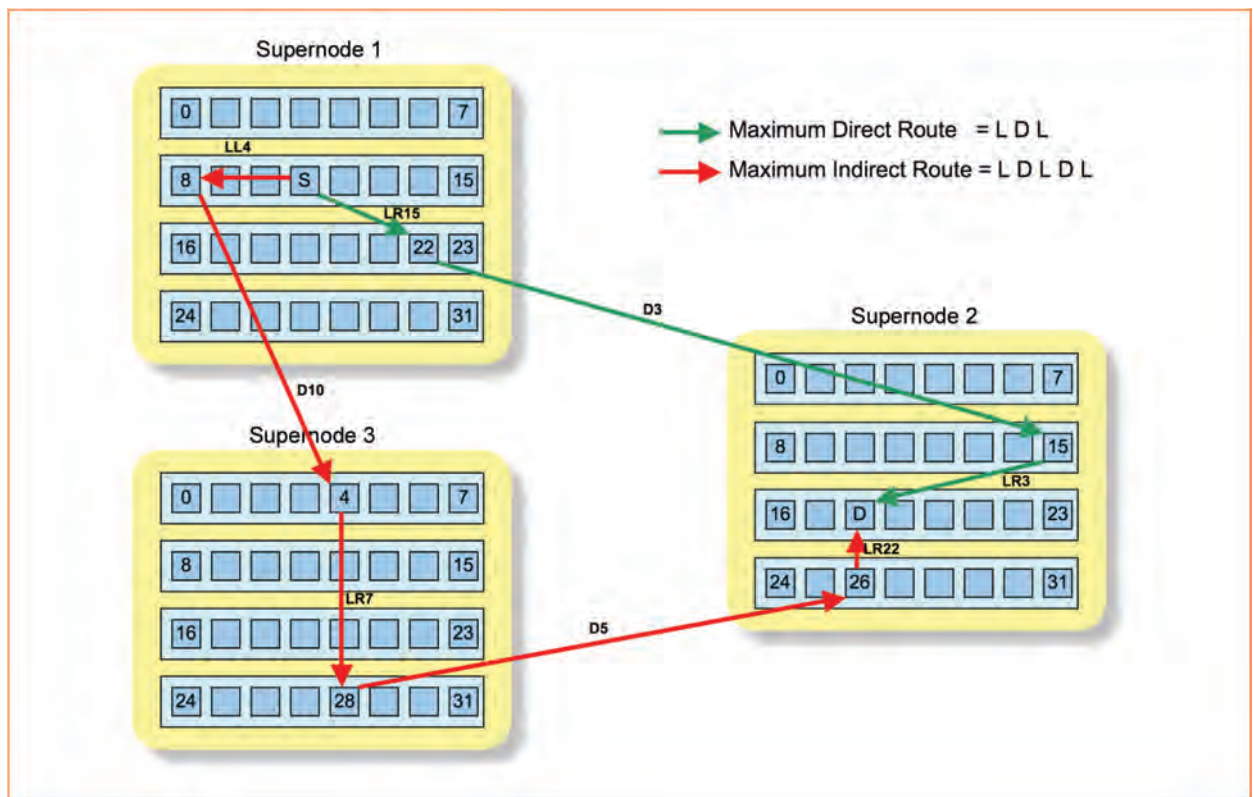


Figure 6

Direct and indirect routes in PERCS.

Figure 7 shows how data flows within the PERCS system. The ISR within the hub chip employs cut-through routing and wormhole routing [17] with 128-byte FLITs. FLITs are assembled into packets, which are the largest units for which the hardware makes an ordering guarantee, i.e., all FLITs of a packet will be delivered in order. No ordering guarantees are provided between packets in a message. Thus, even packets sent from the same source window (see the section that describes the hub chip in detail) and node to the same destination window and node may reach that destination in a different order.

Figure 7 also shows two HFIs cooperating to move data from the POWER7 chips attached to one PowerBus to the POWER7 chips attached to another PowerBus through the interconnect. Note that the path between any two HFIs may be indirect, requiring multiple hops through intermediate ISRs.

In addition to the direct- and indirect-route tables, the ISR also has multicast route tables for replicating and forwarding IP multicast packets. All of the route tables are set up during system initialization by the network management software. In the event of link or other failures, the network management software is alerted and intervenes to reset the tables.

RAS and failure management

The hub chip is architected and designed to support the POWER7 RAS strategy. The main RAS goals are to have no single point of failure and, effectively, 100% error detection on any data paths that could result in data corruption.

The hub chip provides a coherent connection between the four POWER7 chips, as well as between the eight hub chips within a PERCS drawer. Hardware protection is provided for single-bit errors on these interconnects, but any unrecoverable error will *checkstop* the entire drawer to prevent possible data corruption. A drawer failure does not cause a cluster failure. The cluster can continue to operate even if there is a complete hub-chip failure.

The L remote and the D links all use a hardware link-level cyclic redundancy check with link-level retry support for handling transient errors. Link replay buffers in the ISR maintain packet data until a link-level acknowledgment (credit) is received. Spare lanes on all optical links permit graceful degradation due to permanent transceiver failures. The system and the cluster can continue to operate with one or more failed links. Fault isolation registers and control checker

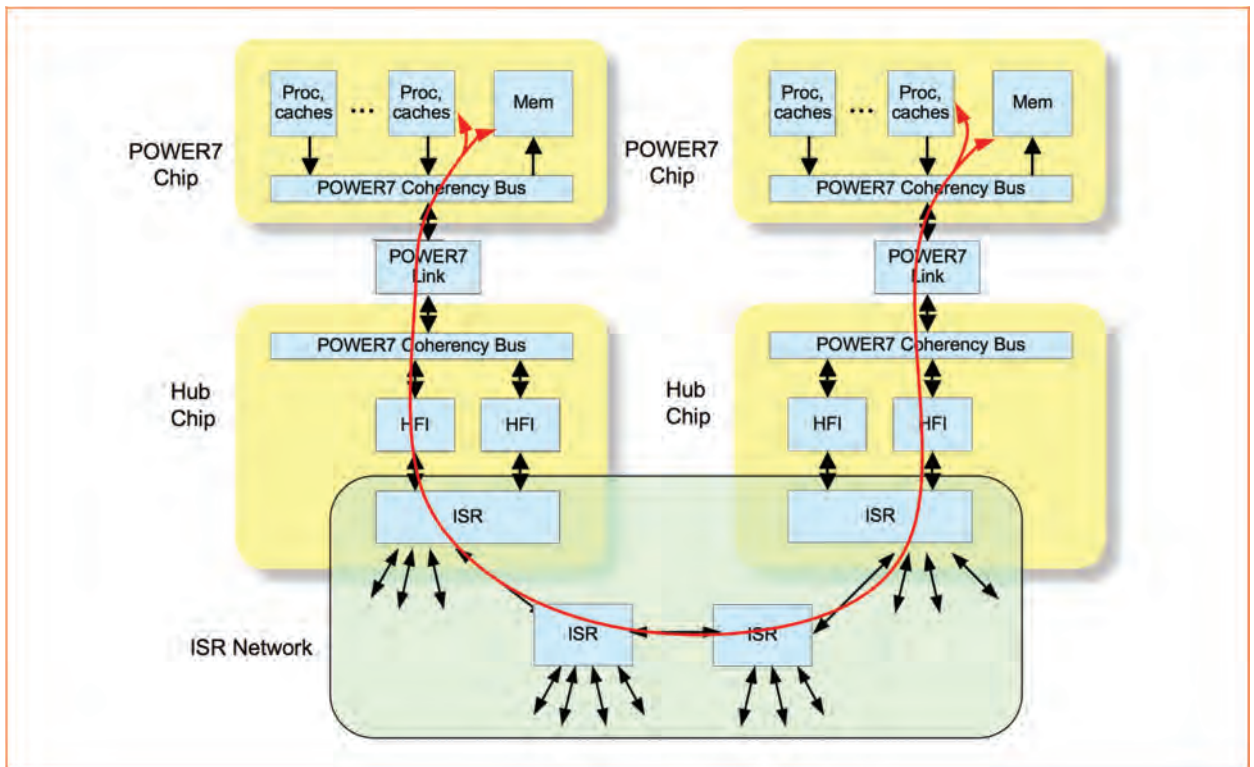


Figure 7

Packet flow in PERCS between two compute nodes. Note that data can both originate from and be written to caches on the source and destination compute nodes.

support permit the hub chip to inform the software of catastrophic failures so that higher level remedial measures can be undertaken. Numerous performance counters, as well as trace debug support, are also provided to assist with both performance and correctness debugging. Finally, multiple indirect routes between any pair of hub chips in the entire system enable not only data striping but also route failover.

Most on-chip random-access memory devices and register files in the hub chip are protected with an error-correcting code permitting recovery from single-bit errors. The only exception to this is in the case of the state for which a failure will only result in a minor and temporary performance (as opposed to correctness) impact (e.g., *least recently used* information).

HPC software

PERCS drawers are configured into multiple resource partitions, each of which runs an operating-system image. Typically, eight partitions will be configured per POWER7-IH drawer such that each partition encompasses four POWER7 chips packaged as a quad-chip module and the hub chip directly connected to the module. Partitions can

be configured to run either AIX* or Linux**. The POWER7-IH drawer firmware provides out-of-band management for the hub chips and coordination with the software, thereby managing the entire interconnect fabric.

AIX and Linux device drivers for the hub chip provide both the Transmission Control Protocol (TCP)/IP network interface and the support for direct user-space access to hub-chip functions. The TCP/IP network interface for the hub chip supports a maximum transmission unit size greater than 60 KB and supports load balancing across multiple links [18].

High-level communication software such as the IBM MPI library [19] is supported by a common low-level API (LAPI) [20], which provides active messages, remote memory copy operations, collective communications, and global atomic memory operations. The LAPI uses a hardware abstraction layer (HAL) to support many different interconnects. The HAL for the hub chip ensures efficient mapping of LAPI functions to the HFI, the global shared memory, and the CAU of the hub chip.

Another high-level software that exploits the LAPI includes a SHMEM** library and a UPC compiler. The UPC

and other PGAS programming models are the primary target for the hub chip's global shared memory facility [21]. The UPC compiler and runtime will generate calls to the LAPI and optimize for the use of functions that efficiently map to the global shared memory, atomics, and CAUs in the hub chip.

In addition to the UPC, IBM XL compilers provide language support for C, C++, and Fortran and OpenMP** [22] directives. Several enhancements have been made to these compilers in support of POWER7 and the PERCS performance and productivity goals. Static and dynamic analysis and assist threads are used to reduce and eliminate cache misses. Polyhedral analysis [23] is used to generate parallel and vectorized instruction streams from serial programs. Transformation reports in an extensible markup language (XML) format are produced to detail areas of the code where compiler optimizations were or were not applied.

Math library support includes both the extended scientific subroutine library (ESSL) and the parallel ESSL (PESSL) [24]. The PESSL uses the MPI to transparently execute math library routines in parallel across the interconnect fabric.

Large-scale parallel applications can be run on the POWER7-IH systems using a jitter mitigation strategy called coscheduling [25]. Coscheduling works in conjunction with the AIX and Linux schedulers and the hub-chip global counter to ensure low-priority tasks run in fixed time slots that are synchronized across all operating-system images. Tasks that cannot be deferred such as interrupts are scheduled to run on symmetric multithreading threads that are otherwise not available for application processing. This strategy ensures that communicating parallel tasks all run together without interruption or jitter for a large percentage of time, avoiding the cascading effects of unsynchronized interruptions.

The software running on utility nodes dispersed throughout a PERCS machine configuration works in conjunction with an executive management server (EMS) to provide overall cluster management and monitoring. Each utility node is responsible for managing a subset of the cluster. The EMS provides a single point of control to the administrator and will delegate tasks to the appropriate utility nodes to be executed in parallel. The EMS and the service nodes are also tied into the hardware control elements. The EMS also runs the software that centralizes the management and the monitoring of the interconnect fabric. The services configured to run on the utility nodes include the Dynamic Host Configuration Protocol, the Network File System, the Extreme Cluster Administration Toolkit (xCAT), and the Resource Monitoring and Control. Compute servers are diskless and rely on their associated service nodes for booting.

The IBM GPFS [10] can be configured to run on all compute, utility, and storage nodes, and uses the TCP/IP to access the hub-chip network interface. Typically, the storage

nodes would be configured as NSD servers, giving all GPFS nodes access to the SAS attached disk enclosures.

LoadLeveler* [26] provides scheduling and resource management for PERCS systems. The resource management component of LoadLeveler works together with the system software and the parallel environment to assign and configure resources needed by a job and manage the execution jobs from launch to termination. LoadLeveler also works with the GPFS to provide a data-aware scheduler, with the xCAT [27] to provide rolling upgrades of the system software and firmware, and with systems management and firmware to manage and reduce energy consumption.

Conclusion

The forthcoming POWER7-IH system known as PERCS is IBM's response to DARPA's challenge to build high-performance highly productive systems. The PERCS hardware employs the POWER7 microprocessor, fully buffered memory, a custom interconnect with a very high fan-out topology, and highly innovative packaging and cooling. Built using robust components such as the LAPI, the GPFS, and LoadLeveler, the PERCS software stack enables users to write applications using not only the MPI but also the UPC and the SHMEM library. Together, the PERCS hardware and software provide a highly productive HPC experience for the user.

Acknowledgments

This material is based on the work supported in part by the Defense Advanced Research Projects Agency under its Agreement HR0011-07-9-0002. This work is also supported by the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation under Award OCI 07-25070 and the state of Illinois. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

*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 PCI-SIG Corporation, Linus Torvalds, Silicon Graphics, Inc., or OpenMP Architecture Review Board in the United States, other countries, or both.

References

1. J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 3rd ed. San Francisco, CA: Morgan Kaufmann, 2003.
2. TOP500, 2006, accessed Jul. 2010. [Online]. Available: <http://www.top500.org/>
3. J. Dongarra and P. Luszczek, "Introduction to the HPC challenge benchmark suite," Univ. Tennessee, Knoxville, TN, Tech. Rep. ICL-UT-05-01, 2005.

4. DARPA Official Press Release, *DARPA Selects Four High Productivity Computing Systems Projects*, Jun. 12, 2002. [Online]. Available: <http://www.darpa.mil/news/2002/hpcs.pdf>
5. DARPA Information Innovation Office, *High Productivity Computing Systems*. Last referenced Dec. 30, 2010. [Online]. Available: <http://www.darpa.mil/i2o/programs/hpcs/hpcs.asp>
6. P. Charles, C. Donawa, K. Ebcioglu, C. Grothoff, A. Kielstra, C. von Praun, V. A. Saraswat, and V. Sarkar, "X10: An object-oriented approach to non-uniform cluster computing," in *Proc. 20th Annu. ACM SIGPLAN Conf. OOPSLA*, 2005, pp. 519–538, DOI: 10.1145/1094811.1094852.
7. B. Sinharoy, R. Kalla, W. J. Starke, H. Q. Lee, R. Cargnoni, J. A. Van Norstrand, B. J. Ronchetti, J. Stuecheli, J. Leenstra, G. L. Guthrie, D. Q. Nguyen, B. Blaner, C. F. Marino, E. Retter, and P. Williams, "IBM POWER7 multicore server processor," *IBM J. Res. & Dev.*, vol. 55, no. 3, Paper 1, 2011.
8. S. Mysore, B. Sinharoy, R. J. Eickemeyer, R. Raghavan, S. Kunkel, T. Chen, W. Maron, D. Flemming, A. Blanchard, P. Seshadri, J. W. Kellington, A. Mericas, A. E. Petruski, V. R. Indukuru, and S. Reyes, "IBM POWER7 performance modeling, verification, and evaluation," *IBM J. Res. & Dev.*, vol. 55, no. 3, Paper 4, 2011.
9. R. X. Arroyo, R. J. Harrington, S. P. Hartman, and T. Nguyen, "IBM POWER7 systems," *IBM J. Res. & Dev.*, vol. 55, no. 3, Paper 2, 2011.
10. F. Schmuck and R. Haskin, "GPFS: A shared-disk file system for large computing clusters," in *Proc. Conf. FAST*, Monterey, CA, 2002, pp. 231–244.
11. PCI-SIG, *PCIe Base 2.1 Specification*. [Online]. Available: www.pcisig.com
12. B. Arimilli, R. Arimilli, V. Chung, S. Clark, W. Denzel, B. Drerup, T. Hoefler, J. Joyner, J. Lewis, and R. Rajamony, "The PERCS high-performance interconnect," in *Proc. IEEE Hot Interconnects*, Aug. 2010, pp. 75–82. [Online]. Available: <http://www.computer.org/portal/web/csd/doi/10.1109/HOTI.2010.16>
13. IBM Corporation, *Power Instruction Set Architecture*, 2009. [Online]. Available: <http://www.power.org/resources/downloads/>
14. MPI Forum, *MPI: A Message-Passing Interface Standard*, Sep. 4, 2009. [Online]. Available: <http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf>
15. U. Consortium, "UPC language specifications, v1.2," Lawrence Berkeley Nat. Lab., Berkeley, CA, Tech. Rep. LBNL-59208, 2005.
16. R. W. Numrich and J. Reid, "Co-array Fortran for parallel programming," *SIGPLAN Fortran Forum*, vol. 17, no. 2, pp. 1–31, 1998, DOI: 10.1145/289918.289920.
17. W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann, 2003.
18. G. Wright and R. Stevens, *TCP/IP Illustrated, Volume 2: The Implementation*. Reading, MA: Addison-Wesley Professional, 1995.
19. W. Gropp, E. Lusk, and A. Skjellum, *Using MPI 2nd Edition: Portable Parallel Programming with the Message Passing Interface*. Cambridge, MA: MIT Press, 1999.
20. D. Quintero, N. Conrad, R. Desjarlais, M.-E. Kahle, J.-H. Kim, H.-N. Nguyen, T. Pirraglia, F. Pizzano, R. Simon, S. L. Yao, and O. Lascu, *Implementing InfiniBand on IBM System p*. Poughkeepsie, NY: IBM Redbooks Publications, 2007. [Online]. Available: <http://www.redbooks.ibm.com/abstracts/sg247351.html>
21. T. El-Ghazawi, W. Carlson, T. Sterling, and K. Yelick, *UPC: Distributed Shared-Memory Programming*. New York: Wiley-Interscience Publishers, 2003.
22. B. Chapman, G. Jost, R. van der Pas, and D. J. Kuck, *Using OpenMP: Portable Shared Memory Parallel Programming*. Cambridge, MA: MIT Press, 2007.
23. C. Bastoul, "Code generation in the polyhedral model is easier than you think," in *Proc. IEEE Int. Conf. Parallel Architecture Compilation Techn.*, 2004, pp. 7–16, DOI: 10.1109/PACT.2004.1342537.
24. Y. Aoyama and J. Nakano, *RS/6000 SP: Practical MPI Programming*. Poughkeepsie, NY: IBM Redbooks Publications, 1999. [Online]. Available: <http://www.redbooks.ibm.com/redbooks/pdfs/sg245380.pdf>
25. S. Seelam, L. Fong, A. Tantawi, J. Lewars, J. Divirgilio, and K. Gildea, "Extreme scale computing: Modeling the impact of system noise in multicore clustered systems," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, 2010, pp. 1–12, DOI: 10.1109/IPDPS.2010.5470398.
26. S. Kannan, M. Roberts, P. Mayes, D. Brelsford, and J. Skovira, *Workload Management With LoadLeveler*. Poughkeepsie, NY: IBM Redbooks Publications, 2001. [Online]. Available: <http://www.redbooks.ibm.com/redbooks/pdfs/sg246038.pdf>
27. L. M. Surhone, M. T. Tennoe, and S. F. Henssonow, Eds., *xCAT*, Betascript Publishing, 2010. [Online]. Available: <http://xcat.sourceforge.net/>

Received October 29, 2010; accepted for publication January 7, 2011

Ramakrishnan Rajamony *IBM Research Division, Austin, TX 78726 USA (rajamony@us.ibm.com)*. Dr. Rajamony received a B.Tech. degree in electrical engineering from the Indian Institute of Technology (IIT), Madras, in 1989, and a Ph.D. degree from Rice University in 1998. He is a Research Staff Member in the Future Systems Group at IBM Research, Austin, and manages the Novel Systems Architecture department there. He has received multiple awards from IBM for his work on high-performance web serving and high-performance computing. He is primarily responsible for PERCS performance and has authored or coauthored more than 20 patents and more than 25 technical papers. Dr. Rajamony is an IBM Master Inventor and a member of the Association for Computing Machinery.

L. Baba Arimilli *IBM Systems and Technology Group, Austin, TX 78758 USA (arimilli@us.ibm.com)*. Mr. Arimilli received an M.S. degree in electrical engineering from Louisiana State University, Baton Rouge, in 1986. He is a Senior Technical Staff Member in the POWER Systems Technology Group at IBM, Austin. He is the Chief Architect of the PERCS High Performance Interconnect and the POWER7 Hub chip. He has held various leadership roles in multiple generations of IBM POWER* processors. In addition, he was the Chief Architect and Director of VLSI Design at an InfiniBand startup company in a previous role. Mr. Arimilli is an IBM Master Inventor and author or coauthor of more than 100 patents and 5 technical papers.

Kevin Gildea *IBM Systems and Technology Group, Poughkeepsie, NY USA (gildea@us.ibm.com)*. Dr. Gildea has a B.S. degree in computer science from the University of Scranton and a Ph.D. degree in computer science from Rensselaer Polytechnic Institute. He is an IBM Distinguished Engineer working on high-performance computing (HPC) software development. He is principally responsible for the DARPA-funded HPCS PERCS project and partnering with NCSA on the Blue Waters project. He has almost 20 years of experience in the development of HPC software and has contributed to the development of the IBM MPI library and underlying communication stack as well as GPFS.

IBM POWER7 performance modeling, verification, and evaluation

In this paper, we describe the key performance enhancements in IBM POWER7® microarchitecture and its memory hierarchy, including performance modeling and verification methodology. We also describe the performance characteristics of server applications, including Standard Performance Evaluation Corporation (SPEC) central processing unit, SAP Sales and Distribution, SPECjbb, online transaction processing workloads, and high-performance computing applications running on POWER7 processor-based systems compared with other systems.

M. Srinivas
B. Sinharoy
R. J. Eickemeyer
R. Raghavan
S. Kunkel
T. Chen
W. Maron
D. Flemming
A. Blanchard
P. Seshadri
J. W. Kellington
A. Mericas
A. E. Petruski
V. R. Indukuru
S. Reyes

Introduction

IBM POWER7* processor-based systems represent unparalleled performance for both transaction processing and throughput workloads. The multicore 45-nm chip design allows up to eight cores per socket and four threads per core, and it delivers a range of performance features that puts POWER7 at the forefront of computing technology. Using the architectural features previously described [1], POWER7 capitalizes on systems designed for performance and flexibility. This paper highlights the performance attributes of the POWER7 product family, including a description of the evolution of the technologies that helped shape the product design.

Design for performance

The architectural design of the POWER7 product family arises from a multitude of intensive modeling efforts, from transistor and deep core simulation to full system modeling. Two important architectural models are discussed in this paper: the core and nest models. The core model focuses on cycle-by-cycle execution in the core and was developed to more deeply investigate the enhanced core features of POWER7. The nest model focuses on the hierarchical effects of data access across all memory components and all of the system domains: core, chip, and node. These models are the cornerstone of the design process for POWER* processors, and they helped shape the final product definitions of POWER7.

Using the models, projection and simulation, the POWER7 engineers were able to sort through the multitude of variables to arrive at the final design. The design was then subjected to another round of design verification, which is also discussed in this paper. Further refinement of the design takes place in the performance verification stage, including the tradeoff analysis of cost, performance, stability, flexibility, interoperability, and schedule. During this time, the performance verification process itself is constantly evolving to ensure the greatest coverage for the latest POWER systems. With increasing complexity and schedule challenges, the verification strategy was adapted to break down the performance verification process to more closely scrutinize the design. Those challenges and lessons are also covered in this paper.

Measuring the performance of POWER7 requires enhanced measurement techniques. This paper discusses enhancements to the performance-monitoring unit (PMU), for example, for cycles per instruction (CPI) analysis and profiling support and thresholding. The PMU in POWER7 has been enhanced to allow precise construction of the CPI stack in addition to being able to profile for every stall condition.

Performance evaluation

The robust performance enhancements of the POWER7 hardware are joined by enhancements in the operating system offerings of POWER Linux** and IBM UNIX** Advanced Interactive Executive (AIX*). These new features are introduced in this paper as well. The software features exploit

Digital Object Identifier: 10.1147/JRD.2011.2147170

©Copyright 2011 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied by any means or distributed royalty free without further permission by computer-based and other information-service systems. Permission to republish any other portion of this paper must be obtained from the Editor.

0018-8646/11/\$5.00 © 2011 IBM

the hardware while offering increased flexibility, stability, and performance. The final portion of this paper focuses on the performance of the POWER7 processor-based systems for a variety of workload configurations. The hardware optimizations unique to POWER7 are highlighted, as well as the software features aimed at POWER7 exploitation. Performance as represented by thread, core, and system in dedicated and shared processor environments is also mapped back to the CPI stack.

Content summary

The first section of this paper describes technology-driven features of POWER7 such as eight cores per chip with embedded dynamic random access memory (eDRAM) Level 3 (L3) private caches, multithread modes, cache enhancements, instruction set architecture-related enhancements such as vector and scalar extension (VSX) architecture, and input/output (I/O) bandwidth enhancements. The second and third sections describe enhancements in AIX and Linux, respectively, to exploit POWER7 core and system architecture capabilities, such as scalability and memory affinity. The fourth section describes in detail the core and the nest modeling that were extensively used in the performance and power/performance design tradeoff decisions. The fifth section describes the performance verification strategies that were adopted.

The sixth section describes PMU features that enable detection of performance bottlenecks in the system. The seventh section documents POWER7 performance characterization work in the context of several workloads such as database, application serving, and Java** in both Max Core and Turbo Core configurations. The features of the POWER7 processor-based systems, including software stack and hardware optimizations, are highlighted in this section, providing examples of enhanced performance in a wide variety of workloads.

Features of POWER7

The IBM POWER7 product family introduces more powerful and more efficient systems that are well suited for a spectrum of workloads to meet the global demand for smarter technology. The POWER7 chip provides four times more cores and eight times more threads, as well as a much improved cache hierarchy latency, compared with the previous generation POWER6*, which translates to more than four times more throughput capability, with higher single-thread performance.

Eight cores per chip

Each POWER7 chip incorporates eight processor cores. Each core, with a dedicated 256-KB Level 2 (L2) and a 4-MB L3, can run four threads, giving it the ability to simultaneously process 32 tasks [1]. The eDRAM technology allows for an on-chip L3 cache, which provides improved

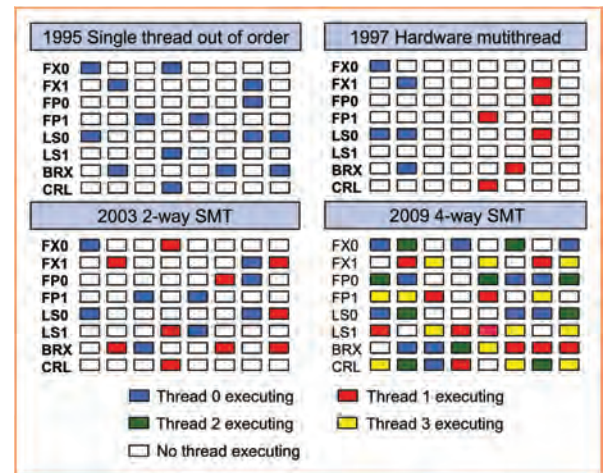


Figure 1

SMT evolution from ST to SMT4.

energy efficiency and reduced latency, resulting in improved throughput.

SMT4

IBM implemented hardware multithreading in the late 1990s [2]. This technology has continued to evolve, resulting in higher compute density per core and improved efficiency (see **Figure 1**) with simultaneous multithreading (SMT).

With a multithreaded design, the core can execute in a single-thread (ST), two-way multithread (SMT2), or four-way multithread (SMT4) mode and dynamically change the mode [3].

To realize the performance gains from SMT support, the flow of instructions for multiple threads is managed to ensure efficient use of resources. When one thread is not making good progress [because of L2 miss, translation look-aside buffer (TLB) miss, sync, or other long latency operations], the other threads will be allowed to have as much of the machine resources as needed for them to make progress. POWER7 controls its instruction flow by giving a large percentage of execution resources to critical tasks. It reduces the amount of resources and power that are used by low-priority work and avoids fine-grain livelock or deadlock situations between threads.

To balance work among the active threads, instruction fetch and instruction decode priorities are reduced for a thread consuming too many resources. The system holds the flow of instructions at the decode pipeline stage for a thread that is going through a long latency event. The dispatch pipeline stage can initiate a “dispatch flush” operation to flush the instructions from a particular thread that are in the instruction pipeline between the instruction fetch and decode

pipe stage to clean any pipeline congestion arising from that thread. If a thread is holding too many pipeline resources and making very little progress because of a long latency event (such as a cache miss), a “balance flush” operation can be initiated, which flushes all instructions from the thread that are younger than the next-to-complete plus one group and holds the refetched instructions from that thread at the instruction buffer stage until the cache miss is resolved. With these design features, POWER7 can achieve very good thread scalability from its SMT4 support and efficiently exploit the workload characteristics.

Cache hierarchy and Turbo Core mode

Each POWER7 core has a private L2 cache and a local L3 cache. Improved eDRAM makes it possible to move the L3 onto the chip for improved energy efficiency and reduced latency, resulting in better performance.

In *Turbo Core* mode, four of the eight POWER7 cores are disabled, whereas their L3 caches are made available to the remaining four cores. In this configuration, the cache of the disabled cores becomes an extension to the local L3 cache in an enhanced caching strategy called Turbo Core mode. With this approach, the local L3 cache per core is effectively doubled in size to 8 MB. In addition, the memory nest is now used by only four cores, which doubles the system bus bandwidth, the memory capacity, and the memory bandwidth per core. Frequency can be boosted in Turbo Core mode by approximately 10% by shifting power from the disabled cores to the active components. These increases in cache, bandwidth, capacity, and frequency in turn improve performance and are intended for the serial components of applications.

VSX advantage

Please see the detailed description in the microarchitecture paper, by Sinharoy et al., in this issue [1].

I/O bandwidth enhancement

The I/O bandwidth capabilities of POWER7 are enhanced compared with those of POWER6, with greater per-socket capabilities that translate into higher overall estimated system bandwidth. **Table 1** shows socket and system bandwidth improvements across midrange (eight-socket system), low-end (four-socket), and entry-level systems (two-socket).

AIX features

IBM offers three operating systems on POWER7: IBM i, Linux, and AIX. This section focuses on AIX improvements to the software stack aimed at exploitation of the POWER7 hardware features.

Capitalizing on the POWER7 hardware are software features provided by IBM’s UNIX AIX Version 6.1 TL5.

Table 1 POWER6 versus POWER7 I/O capability (duplex).

<i>System</i>	<i>I/O bandwidth per socket</i>	<i>I/O bandwidth per system</i>
POWER6 570 vs. POWER7 770 (midrange)	2× improvement	1.7× improvement
POWER6 550 vs. POWER7 750 (low end)	2.7× improvement	2.5× improvement
POWER6 520 vs. POWER7 740 (entry level)	4.1× improvement	4.1× improvement

Scalability support of 1,024 ways

The POWER7 795 model supports a maximum configuration of 256 cores. As each core supports four SMT threads, the AIX operating system is enhanced to support 1,024 threads, treating each thread as a logical central processing unit (CPU) core. Changes have been made to all operating system structures and algorithms to enable a fully scalable hardware and software solution for POWER7.

Enhanced affinity

User data and home node

In POWER6, all memory in the system was striped across memory domains. The memory was evenly distributed per POWER chip. POWER7 includes an enhancement to memory affinity that allows memory associated with user data, such as the stack, the heap, and certain private kernel structures, to be kept close to the cores that are processing the data; that is, each thread has a “home SRAD,” or scheduling resource affinity domain, and the user data is associated with the home SRAD. Local references for the data items reduce latency and fabric traffic and improve application performance. Enhanced Affinity is an operating system feature, enabled by default, on AIX 6.1 TL5.

When the system is very busy, the Enhanced Affinity keeps user data close to the core. When the system is not busy, the AIX scheduling load balancers will try to migrate threads to less busy SRADs. To avoid unnecessary latencies, the schedulers will try to schedule another thread on the same core. If there are no additional threads, the scheduler will look for a thread on the same chip. If no chip-level thread is available, a thread from the same node will eventually be chosen.

Single-threaded applications, such as some aspects of online transaction processing (OLTP) workloads, are well suited for Enhanced Affinity. Multithreaded workloads can still take advantage of affinity through the existing kernel-level affinity that was made available in POWER6. The kernel-level affinity is provided via dispatch algorithms designed to keep threads running on cores to which they were

previously dispatched. The kernel-level affinity provided in POWER6 did not use the affinity information about the partition's physical memory location in its dispatch algorithm. This was addressed in the POWER7 Enhanced Affinity.

Dispatch affinity

POWER7 introduces new functionality that improves shared processor logical partition (SPLPAR) dispatch affinity. Partitions running in SPLPAR mode can still run on multiple POWER7 cores, but cores that are preferred because of their affinity characteristics are targeted for dispatch. The POWER Hypervisor* on POWER7 dispatches SPLPAR partitions more frequently on cores that are close to the partition's physical memory allocation, thus reducing memory access latencies. Under strenuous conditions, SPLPAR partitions may be forced to run on cores that are remote from the partition's physical memory, but the POWER Hypervisor will dispatch the partition back to the preferred core as soon as processing cycles are available. The changes made to the POWER7 architecture result in even lower virtualization overhead in an SPLPAR environment.

POWER6 compatibility mode

POWER7 supports a POWER6 compatibility mode. This attribute is aimed at providing a more flexible partition mobility option for partitions that have been configured to run only with the POWER6 instruction set. Using the POWER6 compatibility mode, the partitions can now use Live Partition Mobility to move to the new POWER7 platform while maintaining strict adherence to the POWER6 instruction set. In POWER6 compatibility mode, POWER7-specific instructions are not enabled, and the SMT mode is set to SMT2 for this partition. In the SPLPAR mode, a pool of processors may be shared across multiple logical partitions. Even in POWER6 compatibility mode, partitions that are configured for SPLPAR mode can still use cores from the general shared pool on the POWER7 platform, sharing them with partitions not running in POWER6 compatibility mode. The POWER Hypervisor will detect at dispatch time whether a partition is running in POWER6 compatibility mode and will quiesce two of the SMT threads.

Linux POWER7 features

A number of Linux kernel modifications support new POWER7 features, including SMT4 and VSX, which contributed to a significant number of Linux- and industry-leading benchmarks.

SMT4

Previously, POWER Linux supported at most two threads per core. POWER7 Linux now supports an arbitrary number of threads per core and can dynamically adjust the number of

threads per core at runtime on a per-core basis. This approach provides a single-threaded workload maximum throughput from its core while simultaneously enabling multithreaded workloads to use all four threads on each remaining core.

VSX

Support for VSX improves floating-point throughput via autovectorization in the GNU Compiler Collection compiler.

POWER7 Linux 1,024-thread scalability

A number of changes were made to Linux in order to support the largest POWER7 systems.

Read-copy update

Read-copy update (RCU) [4] is an IBM-contributed technology that improves scalability and performance, primarily by replacing reader-writer locking. RCU read-side primitives incur extremely low overhead, incurring zero overhead in server-class builds of the Linux kernel. RCU updates do not exclude readers, and thus, must wait until all preexisting RCU readers finish before undertaking destructive operations, such as freeing memory. Using a combining tree, a wait-for-readers operation is available in the most recent implementations of RCU. This enables RCU to efficiently operate across the full range of POWER7 hardware scalability.

Memory affinity

Previous versions of the POWER Linux kernel supported a maximum of 16 memory domains. POWER7 Linux supports 256 memory domains, which is more than sufficient for the up to 32 domains on POWER7 systems. Where possible, Linux allocates all process memory from the local memory domain, and the scheduler tries to schedule a task within the domain from which it last ran. This approach reduces average memory latency, thus improving both performance and scalability.

Performance modeling

This section describes the model and modeling process. Performance models are a fundamental part of the design process to understand how the design will behave. The basic uses of a performance model are to guide design tradeoffs, to project performance, and to aid in code tuning. For design tradeoffs, a model can compare alternative designs (whether vastly different designs or just fine-tuning one aspect of a design) and can identify bottlenecks that lead to design changes. Models contribute to both performance tradeoffs and power tradeoffs. For projections, the model is used to produce the expected CPI based on the model details and inputs to the model. When combined with frequency and path length, and in some cases, accounting for memory capacity constraints, the throughput is determined. Code tuning allows running different sequences of instructions and

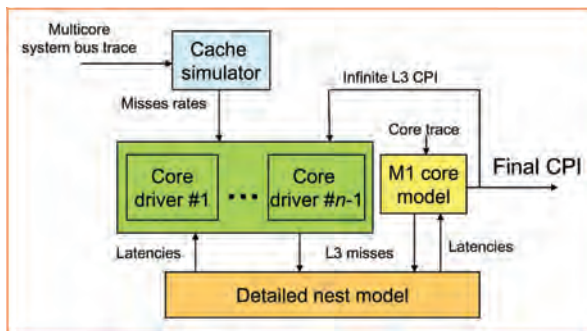


Figure 2

Multicore multichip simulation methodology.

observing the behavior, both at a high level (CPI) and at a low level through a visual representation of execution. This section describes the basics of the model, the generation of inputs (traces and probabilities), and the model validation.

A detailed cycle-by-cycle model of the core and nest exists, which is trace driven with some probabilities used in certain modes. The model is known as M1. The model can be run in several different modes. One mode is a single core and associated nest without any other cores in use. The L3 cache associated with the modeled core is modeled as is the memory and associated buses. The L3 caches associated with other cores can be optionally enabled to service the operational core. All caches are deterministic. A second mode of operation is to run one core and have $N-1$ simple core drivers generating activity with a detailed nest model to fully simulate an N -way system. This is shown in **Figure 2**. The simple driver cores generate nest requests at a rate that matches that of the simulated core by observing the CPI of the simulated core. Probabilities derived from cache simulations using bus traces from previous generations of servers, drive the cache miss rates, sources of data, and coherence states such that a full N -way system is simulated. A third mode is to remove the full core model and now use N simple driver cores. This also simulates a full N -way system but with faster runtime. The simple core request rate is obtained from the $N-1$ setup, possibly extrapolated due to different system configurations. A fourth mode is similar to the $N-1$ mode but is run with deterministic caches. Here, each simple core watches and echoes nest requests from the modeled core, with some offset in address and offset in time. This is used for workloads not sharing data, such as Standard Performance Evaluation Corporation (SPEC) rate.

Core modeling

The model is split into a core and a nest component with each assigned to a different team. The two parts of the model are independently updated at the rate deemed appropriate, as

each team works with the respective design team. The core model is mostly concerned with the single-processor aspects of the design, whereas the nest model is mostly concerned with multiprocessor aspects of the design.

The core model consists of a detailed cycle-by-cycle model of the execution pipelines, out-of-order tracking mechanisms including queues, rejects, and flushes, Level 1 (L1) and L2 caches, translation facilities, instruction fetching, and branch prediction. Over the course of the design, hundreds of design studies are performed on a wide variety of workloads. The model produces a large amount of statistical data to help predict problems or simply compare contrasting settings. Outputs are saved in a database to facilitate comparisons and design history. The model implements many of the performance monitor counters (PMCs) maintained in the hardware to enable model validation, comparing model output and hardware behavior. In addition, the model also produces a large number of statistics, which are inputs to a power model used for projecting power usage. There are also capabilities for looking at power and switching at a fine-grain level. For debugging and software tuning, the model produces a cycle-by-cycle picture of execution, illustrating where and why instructions are stalled, for example. When coupled with an architectural simulator, it becomes easy to execute a code and observe the performance results on a cycle-by-cycle basis.

Nest modeling

The nest is a hierarchy of interconnects among the cores within a chip, the chips within a node, and the nodes within a bigger system. The intent of the nest model is to properly account for the latency for a miss at the L3 cache in the core. When a miss occurs at the L3 cache in the POWER7 core, the model knows *a priori* where the instruction or data is going to come from. In the simplest case, an application running on a single core would directly get the instruction or data from memory private to that chip, in the event of the L3 cache miss. In the other extreme case, where an application runs as a single image on the entire system consisting of multiple nodes, the source of the data could be potentially any cache or memory in the system. In the latter case, the nest model allows a CPI impact estimation of instruction and data sharing among all the cores in the system.

When multiple cores are running as a single system image, several instruction and data transfers could be potentially crisscrossing the system nest at the same time. The nest model accounts not only for the actual wiring and logic delays for each transfer from source to destination but also for all the queuing delays along the way. By modeling the shared resources in the fabric such as the arbiters and the buffers, the nest model allows high-level optimization during the design phase of the various algorithms, buffer sizes, and critical latches so that few bottlenecks arise in the system.

One critical component of the nest model is the two on-chip memory controllers and the dual in-line memory modules (DIMMs) that are connected to each chip. The nest model reflects the interleaving of addresses among the controllers and DIMMs so that when bank or rank conflicts occur, the queuing delays that result are properly accounted for. Various algorithms used in the memory controller, such as the read and write request dispatch logic, the command throttling, and the rank power-down algorithms, the DRAM refresh logic, among others are also modeled in detail. These details ensure that the memory latency is within reasonable limits, and the memory bandwidth is adequate for a wide variety of applications.

Model inputs

A fundamental input to the model is the trace. Traces to drive the model come in two forms: instruction and bus traces. Instruction traces contain instructions, instruction and data addresses, and taken branch information. These drive the core model. Bus traces contain addresses of different types from a multiprocessor system. These drive the nest model both with and without the use of the core model. Instruction traces and bus traces can be collected from special hardware, similar to that described in [5]. In addition, instruction traces can be generated from a full-system simulator. This is described in more detail below. For hardware collection, instruction traces include the instruction image and instruction and data addresses. It is possible to simultaneously collect traces from more than one core to provide a set of simultaneous traces. Hardware traces are analyzed, and a subset from each workload is selected for input to the model. The selection is based on determining how representative that subset is of the entire workload and comparing hardware measurements of a workload against a simulation of the same hardware. The selected traces are typically contiguous traces, which are not sampled. Multiuser commercial workloads typically do not have distinct phases, such as in a technical workload, as the many users are each running mostly independently, and their activities blend together.

For SMT simulation, a single trace is split into multiple threads, or a simultaneous collection set is used. Splitting a single trace provides more flexibility as any number of threads can be generated, within reason, without having to know upfront how many threads to collect.

Bus traces are also collected from hardware. Long traces can be collected to simulate large caches. Traces on existing hardware are limited in the number of processors that can be collected by the size of the hardware. By analyzing various components of the miss rates, extrapolations to more processors can be done. Other adjustments to miss rates can be made for a variety of reasons, including affinity to processors or nodes. Some of these adjustments have been previously described [5].

Performance verification

The POWER7 processor is the most advanced and complex POWER processor ever built. It has twice the number of transistors, four times the cores, and eight times the threads of the predecessor POWER6 processor. As the complexity of a processor increases, so does the complexity of verifying the function and performance of the processor. Added features increase the number of tests that have to be run, and the complexity compounds this by increasing the number of variables (scenarios) needed to assure coverage for those tests. Traditional verification methodology would require a longer schedule to accomplish these tests, but POWER7 actually had a shorter schedule, compared with previous projects. Thus, POWER7 presented an unprecedented performance verification challenge, both in complexity and in time. This section describes the various performance verification strategies that were devised to meet this challenge.

Performance verification strategy overview

The performance verification of POWER7 entirely focused on clock-cycle-based measurements. Frequency estimation was separately handled through a combination of analog simulation and static timing analysis. Some items for which exhaustive coverage in simulation would be prohibitive [e.g., cache least recently used algorithms] were verified via formal methods.

Performance verification tests, which are called performance verification programs (PVPs), were run in a hierarchical manner. The process is started with low-level tests that exercise individual core and memory nest (cache + memory) units, then to higher level tests that exercised either the entire core or the entire nest model. Core tests were mainly written in assembly language, some by hand, and some compiled from C and Fortran application code. Nest tests were not written in assembly code but as text files that a software “core driver” would read in order to generate sequences of bus requests (e.g., loads, stores, instruction fetches) that would be sent to the nest. All of the unit-, core-, and nest-level PVPs were run on (Very-high-speed integrated circuit Hardware Description Language (VHDL)) models that had been compiled to run in Mesa, which is a cycle-based logic simulation tool that runs on a standard Linux/AIX server.

To determine “pass/fail” status of each PVP, the results were compared either against performance targets that had been established by designers or against the results for the same PVP on the M1, our cycle-accurate software model of the core and the nest. The “pass” criteria of each PVP varied depending on what the test exercised. Some tests targeted specific design features, for example, the branch predictor. These feature-specific tests would typically rely on specific event counts, such as, in this example, mispredicted branches, to indicate success or failure of the test. Other tests were aimed toward recreating small samples of code

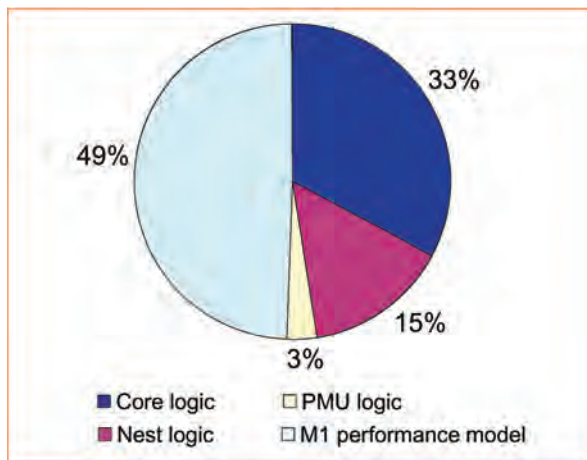


Figure 3

Performance bugs found in POWER7 performance verification.

commonly found in applications or compiler generated constructs. These typically relied on a CPI comparison versus the M1 performance model to indicate success or failure. In the nest model, results were typically compared against various latency and bandwidth targets.

The third and highest level of tests was the chip-level tests, in which long code sequences from various benchmarks and workloads, mostly certain SPEC CPU2006 benchmarks and some high-performance computing code kernels, were run on chip level VHDL models and the M1. The “pass/fail” criterion was entirely determined by CPI. The chip-level tests were primarily simulated on the Awan family of VHDL model accelerators (described later), which had never been so extensively used for performance verification. Due to these chip level tests, two months were cut from the POWER7 schedule, in an effort known as “shift left.”

Altogether, the three-level performance verification strategy uncovered hundreds of performance bugs. **Figure 3** shows a breakdown of the various performance bugs found in the POWER7 performance verification process. The bugs were split almost equally between the M1 performance model and the VHDL. Of the bugs found in VHDL, approximately two-thirds were in the core and one-third in the nest.

Shifting the schedule left: Role of Awan VHDL accelerators in accelerating POWER7 development

At the beginning of the POWER7 project, the performance and verification teams started an informal project to use an architectural simulator called Systemsim to develop the tools and test cases that would be eventually run on early POWER7 hardware. A secondary goal was to use hardware

VHDL emulation (Awan) as an early bring-up environment and run a few of the tools and test cases on actual VHDL. Under schedule pressure, the teams extended their objective to more comprehensive performance testing on Awan in order to shorten the time spent in the laboratory once parts arrived. Samples of important workloads were executed, including a subset of the SPEC CPU2006 benchmarks and some high-performance computing code kernels, in order to exercise the design in a way similar to the laboratory environment. The result was a dramatic improvement in pretape-out coverage, a reduction in hardware bugs, and at least two months taken out of the hardware test schedule.

Description of Awan performance verification environment

For POWER7, a new simulation infrastructure and methodology, which is illustrated in **Figure 4**, was created to allow for the comparison of code running on a hardware accelerated VHDL (Awan) simulation model to the M1 software performance model. The goal of this effort was to identify differences between the two models in order to identify bugs in the VHDL and in the M1 itself.

This environment begins with the UltraPoint sampling methodology to identify samples of execution of a given workload that fairly represent the performance of the entire workload. When running benchmarks and real-life workloads on Awan or M1, sampling is necessary because the dynamic instruction path length of each benchmark or workload can reach into the trillions, which would take the typical simulator years to execute. To run the models in a reasonable time frame, the sample size of each benchmark is limited to 100 million instructions, with up to 200 million additional instructions for warming of structures such as caches and branch predictors. This ensures that the benchmark runs would complete in 1–2 days. The UltraPoint methodology is based on the SimPoint clustering methodology [5], with some changes for greater accuracy and to better suit the modeling environment.

The samples chosen by the UltraPoint methodology are represented as counts that identify the number of instructions in the sample and the number of instructions that must be executed to get to the beginning of the sample from the end of the previous sample. This data is used to rerun the binary on the Systemsim architectural simulator to create instruction traces for the M1 model, which is trace driven, and architectural checkpoints (values of memory and architected registers) for the Awan model, which is execution driven.

The traces are run on both models, and performance data is collected, including CPI, cache miss rates, and average memory latency. The performance data from each model is compared to isolate and fix any discrepancies in either model.

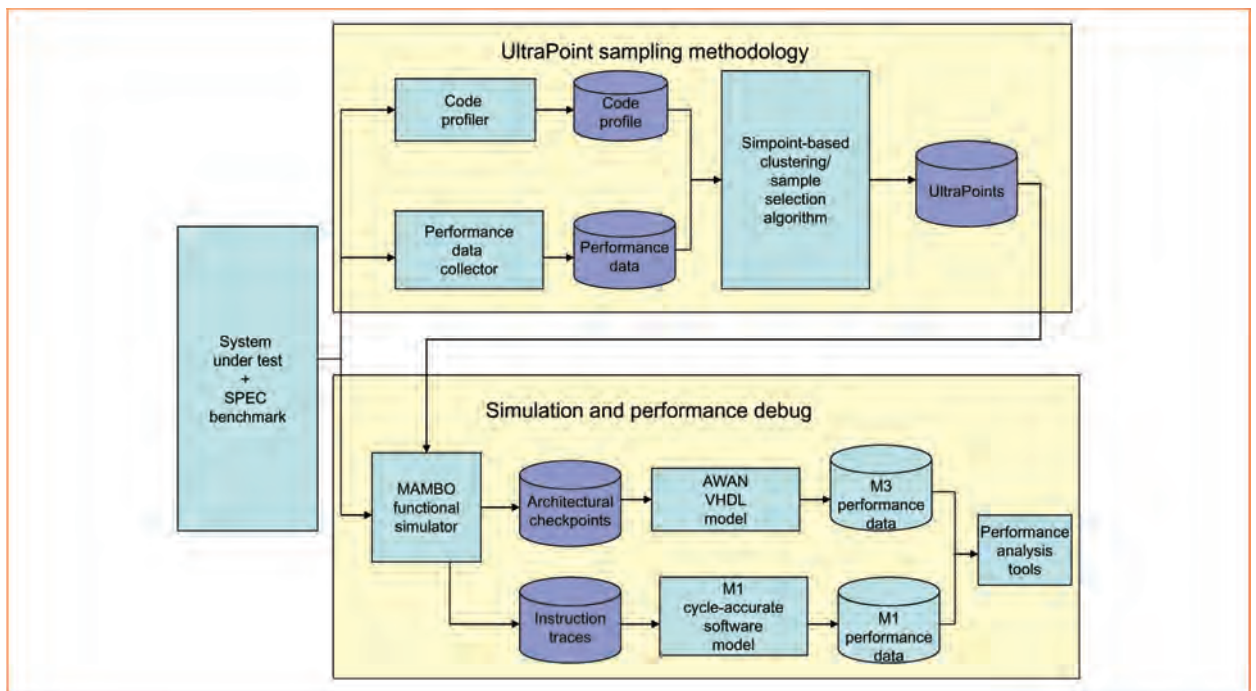


Figure 4

Awan performance verification methodology.

Bugs discovered in Awan performance verification

Several significant bugs were uncovered and fixed during Awan performance verification, including not only bugs in the VHDL but also in the M1 cycle-accurate model. Several of these bugs could not have been found in the standard Mesa software VHDL simulator since Awan's speed allows many more instructions to be simulated, exercising the VHDL in ways that could only be previously done in the laboratory.

More PMU bugs were found than any other kind of bugs (such as Awan, non-PMU performance bugs, and M1 model bugs). The PMU allows the user to collect various performance measurements on hardware and is used by the performance team to validate the system performance. Since the PMU must correctly work in order to validate any aspect of performance, the first several months of hardware performance testing in the laboratory are spent running increasingly complex performance tests to determine which PMU features properly work. Only after this laboratory testing can the hardware be turned over to the software performance team to begin their tuning effort. The software performance team also heavily relies on PMU output to tune software features; thus, the work of validation on the PMU was of critical importance.

In previous projects, many of the PMU's events or features could be fully exercised only after hardware was available.

PMU bugs found during this early laboratory testing often slow down both hardware and software tuning efforts, often until subsequent (and very expensive) chip tapeouts fix the problem. Bugs found later are even more disruptive since the PMU data can be misinterpreted in a way that hides the actual source of performance concerns.

Awan verification also uncovered many non-PMU performance bugs in the VHDL. Several of these bugs, if not found on Awan, would only have been found in the laboratory and, if found late, may have had an impact on the schedule, which could have had a severe impact on POWER7 competitiveness and profitability.

The Awan work also drove a significant number of fixes and configuration changes to the detailed processor performance model, i.e., the M1. Like the VHDL bugs, several of these fixes would not have been driven by the standard Mesa/PVP environment because of the lower complexity of that environment. As the M1 model is used to project performance on key benchmarks and workloads, the Awan work also led to more accurate performance projections.

Together with the traditional Mesa VHDL simulator, the Awan VHDL accelerator provided the coverage and the speed necessary to deliver a processor with unprecedented performance capability on a historically short schedule.

Performance monitoring

The POWER7 processor has a built-in PMU, which is designed to provide instrumentation to aid in performance monitoring, workload characterization and system characterization code analysis. The PMU comprises six thread-level PMCs. PMC1–PMC4 are programmable. PMC5 counts nonidle completed instructions, and PMC6 counts nonidle cycles. Each processor core also has a core-level facility called hypervisor PMC (HPMC), which aggregates counts from all the threads for a few selected events. There are four HPMCs per core. HPMC1 and HPMC2 are configurable, HPMC3 counts aggregate nonidle completed instructions, and HPMC4 counts nonidle core cycles.

The thread-level and core instrumentation has access to a rich set of performance events (close to 530) that covers essential statistics such as miss rates, unit utilization, thread balance, hazard conditions, translation-related misses, stall analysis, instruction mix, L1 instruction cache and data cache reload source, and effective cache and memory latency counts. Apart from the thread and core-level performance counters, POWER7 also has support for chip-level instrumentation via self-contained PMUs called PMUlets. Several PMUlets are distributed across the chip to provide insight into chip-level metrics such as memory bandwidth, PowerBus utilization, and I/O bandwidth. Each PMUlet has a 64-bit configuration register and four effective 32-bit counters.

The following sections provide an overview of some of the advanced PMU features implemented in the POWER7 processor.

CPI analysis and stall accounting

CPI is the fundamental metric in processor performance. Knowing the number of cycles (or fractions of a cycle) it takes to complete an instruction is critical to evaluating the performance of the processor. CPI can be broadly broken down into three major components:

- Cycles a thread used to complete a group of instructions.
- Cycles in which a thread was starved for work.
- Cycles during which a thread was stalled.

CPI analysis is complicated on the POWER7 processor because of the speculative out-of-order execution of the core; in a given clock cycle, there could be 120 internal operations in flight, 20 groups of instructions past dispatch, 32 outstanding loads/stores, 16 outstanding loads to 8 independent L1 cache line misses, and 4 independent threads. With up to six internal operations per group, any one of the instructions could block completion, and multiple conditions could block any instructions (a translation miss followed by a cache miss, for example). Even if stall conditions could be measured on a per-instruction (or even per-group) basis, that information by itself is not sufficient

to break down the completion stall component because speculation might cause the instructions to be discarded before they complete. Until a stall condition has resolved, it might not be possible to know its fundamental cause.

POWER7 aids the analysis of CPI breakdown by implementing speculative, or rewind, counters. Two of the programmable counters have backup registers that are not accessible to software. When software writes an initial value to the counter, its backup register is also written. A counter can be configured for a particular stall condition and begins speculatively counting on any cycle when no group is completing. The first group that completes will report the last condition that held its completion. If the condition matches what the counter is configured for, the count value is committed by updating the backup register. If it does not match, the counter is rewound to its previously check-pointed value.

Figure 5 shows the hierarchical breakdown of cycles possible on the POWER7 processor.

The events are hierarchical. Load/store unit (LSU) stalls will include data cache miss and rejects. Reject stalls will include translation stalls due to effective-to-real-address translation (ERAT) misses. This method only reports the last condition to clear and will not detect dependence chains. For example, a fixed-point *add* instruction that is dependent on a load in the same group that misses the data cache will cause the completion stall condition to be reported as a fixed-point unit (FXU). When multiple conditions are reported at the same time, load/store conditions are favored over all others, and data cache miss is favored over rejects.

Completion can be also stalled because there are no instructions in the pipeline. The primary causes of this are instruction cache misses and branch mispredicts. POWER7 has speculative events for these conditions that are handled in the same way as completion stalls.

Enhanced profiling support

Profiling or sampling is a common approach to associate expensive performance events in a processor to instruction and data addresses. Profiling allows you to identify hotspots in code and data, find performance-sensitive areas, and identify problem instructions, data areas, or both. Profiling is commonly achieved by identifying a particular instruction and collecting detailed information about that instruction (instruction sampling). The PowerPC* architecture provides two special-purpose registers to identify sampled instructions. The sampled instruction address register (SIAR) captures the effective address of the sampled instruction. The sampled data address register (SDAR) captures the effective address of the sampled instruction's data operand, if any. An indicator bit in monitor mode control register (MMCRA) indicates when the SIAR and SDAR are from the same instruction. The SDAR is not cleared when a new sampled instruction is selected; therefore, the indicator bit is needed to show that the SDAR is not for a previously (since executed or cancelled) sampled

Cycles (PM_RUN_CYC)	Completion Stall Cycles <C> (PM_CMPLU_STALL)	Stall by FXU <C2> (PM_CMPLU_STALL_FXU)	FXU Multi-Cycle Instruction <C2A> (PM_CMPLU_STALL_DIV)	
			FXU Other <C2 - C2A> (PM_CMPLU_STALL_FXU_OTHER)	
		Stall by VSU <C3> (C3A +C3B +C3C) (PM_CMPLU_STALL_VSU)	Stall By Scalar <C3C> (PM_CMPLU_STALL_SCALAR)	Stall By Scalar Long <C3C1> (PM_CMPLU_STALL_SCALAR_LONG)
				Stall By Scalar Other <C3C2: C3C-C3C1> (PM_CMPLU_STALL_SCALAR_OTHER)
			Stall by Vector <C3B> (PM_CMPLU_STALL_VECTOR)	Stall by Vector Long <C3B1> (PM_CMPLU_STALL_VECTOR_LONG)
				Stall By Vector Other <C3B2: C3B-C3B1> (PM_CMPLU_STALL_VECTOR_OTHER)
		Stall by DFU <C3A> (PM_CMPLU_STALL_DFP)		
		Stall by LSU <C1> (PM_CMPLU_STALL_LSU)	Stall by Reject <C1A> (PM_CMPLU_STALL_REJECT)	Translation Stall <C1A1> (PM_CMPLU_STALL_ERAT_MISS)
				Other Reject <C1A2: C1A - C1A1> (PM_CMPLU_STALL_REJECT_OTHER)
			Stall by D-Cache Miss <C1B> (PM_CMPLU_STALL_DCACHE_MISS)	
			Stall by Store <C1C> (PM_CMPLU_STALL_STORE)	
			LSU Other <C1C: C1 - C1A - C1B - C1C> (PM_CMPLU_STALL_LSU_OTHER)	
		Stall due to SMT <C4> (PM_CMPLU_STALL_THRD)		
		Stall due to IFU <C5> (PM_CMPLU_STALL_IFU)	Stall due to BRU <C5A> (PM_CMPLU_STALL_BRU)	
			Other IFU Stall <C5B: C5 - C5A> (PM_CMPLU_STALL_IFU_OTHER)	
	Other Stall <C6: C - C1 - C2 -C3 - C4 - C5> (PM_CMPLU_STALL_OTHER)			
GCT Empty Cycles (PM_GCT_NOSLOT_CYC)	GCT Empty due to lcache Miss <B1> (PM_GCT_NOSLOT_IC_MISS)			
	GCT Empty due to Branch Mispredict <B2> (PM_GCT_NOSLOT_BR_MPRED)			
	GCT Empty due Branch MisPrdict and lcache Miss <B3> (PM_GCT_NOSLOT_BR_MPRED_IC_MISS)			
	GCT Empty other <B - B1 - B2 - B3> (PM_GCT_EMPTY_OTHER)			
Completion Cycles (Group Completed) <A> (PM_GRP_CMPL)	Base Completion Cycles <A1> (PM_1PLUS_PPC_CMPL)			
	Overhead of expansion <A - A1>			

Figure 5

CPI stack legend.

instruction. The sampled registers can only be updated by the processor when performance monitor exceptions are enabled. Performance monitor exceptions toggle the enable bit, locking the contents of the sampled registers. This makes it possible to profile code not enabled for interrupts.

Some sampling techniques include the following:

- *Event-based sampling*—Sampling an instruction based on the occurrence of an event, such as a cache miss.
- *Continuous sampling*—Sampling every instruction executed.
- *Slot-based sampling*—Sample instructions based on some uniform criteria such as position in an internal queue.
- *Random sampling*—Randomly sample instructions.

The POWER7 processor supports continuous sampling and random-instruction-based sampling. With random instruction

sampling, an instruction is randomly marked at dispatch, and the performance events that the marked instruction generates are routed to the PMCs. The SIAR and SDAR precisely capture the instruction and data effective address of the marked instruction. This technique is extremely useful for tying a single performance event to a precise instruction or data address.

While this technique is extremely useful, processors since POWER6 have had the ability to associate multiple performance events to the same instruction or data address. This is accomplished by recording up to 32 events that pertain to a marked instruction during the lifetime of the marked instruction in the pipeline in a software accessible register (MMCRA [0:31]).

The following types of information are recorded:

1. Cache miss indication and reload information.
2. TLB miss and reload information.

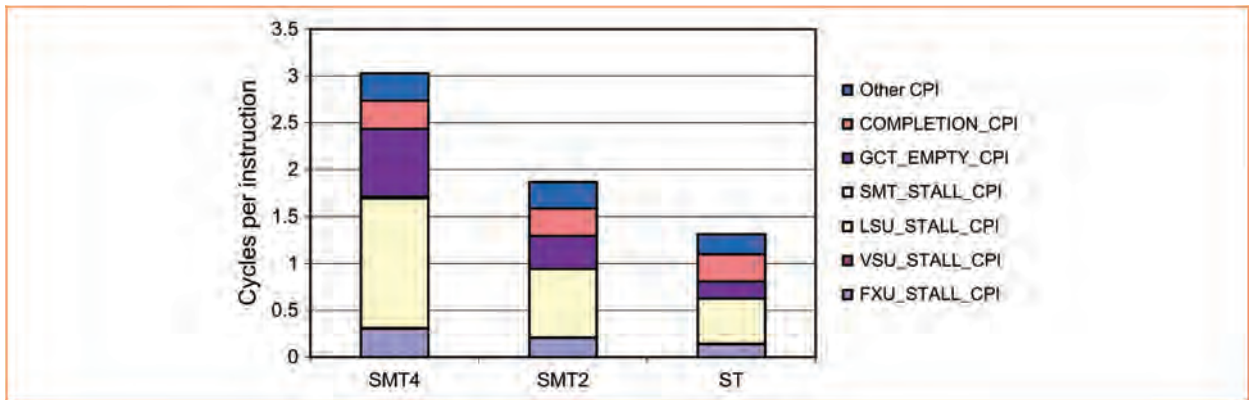


Figure 6

SAP SD CPI components (per thread).

3. Load/store reject indication.
4. Program state information (problem/supervisor/Hypervisor).
5. Branch mispredict indication and branch taken indication.
6. SIAR/SDAR correlation indication.

With this capability, it is possible to collect multiple pieces of information about an instruction/data with just one pass through the program. As an example, profilers can find whether an instruction or data address both suffered a translation or cache miss, and if so, at which level or cache or memory did the miss resolve.

POWER7 enhances this capability by augmenting MMCRA [0:31] with the following CPI stall indication associated with a marked instruction.

1. *Marked instruction suffered a finish stall*—This means that the marked instruction finished after the group containing the marked instruction was next to complete. If this indication is not set, the marked instruction experienced out of order benefit.
2. *Marked group suffered a completion stall*—This means that the group containing the marked instruction suffered a completion stall.
3. *Completion stall reason*—If the group experienced a completion stall, the stall reason is encoded as a part of the indication.

While CPI stack provides a hierarchical breakdown of cycles, this profiling capability allows software programs to exactly identify those sections of code that suffer from completion stalls.

Thresholding

As with past PMU designs, the POWER7 PMU design provides the ability to obtain counts of the number of marked

(sampled) instructions for which the execution time between a designated start/end pipeline event pair exceeds a specified threshold value. To support this, the processor provides the ability to program a threshold value and specify start/stop pairs. POWER7 provides the following start/stop pairs for marked instructions:

Instruction dispatched.
 Instruction issued.
 Instruction finished.
 Instruction completed.
 Instruction suffered a D-cache miss.
 D-cache reload for a marked instruction.
 Instruction is next to complete.

Only one marked instruction is active in the pipeline at a time, and only one threshold value can be used for comparison with the selected start/end event pair. Once a pair of start/end threshold events is selected and the start event occurs, the threshold facility will begin decrementing from the threshold value and will continue to decrement until either the decrementer times out, another start event occurs, or the end event occurs. If the decrementer times out, and if the threshold logic event is selected for counting, the threshold logic event counter is incremented, and indication that threshold exceeded is set in MMCRA [0:31].

Bringing the advanced features together

This section provides an illustration of how the advanced features of the POWER7 PMU can be used to identify load misses that are typically responsible for very long execution stalls. POWER7 has the capability by enabling all the advanced features to tie together the following pieces of information for a load miss into a quadruple.

Profilers can enable random instruction sampling and select start and stop events for thresholding as marked cache

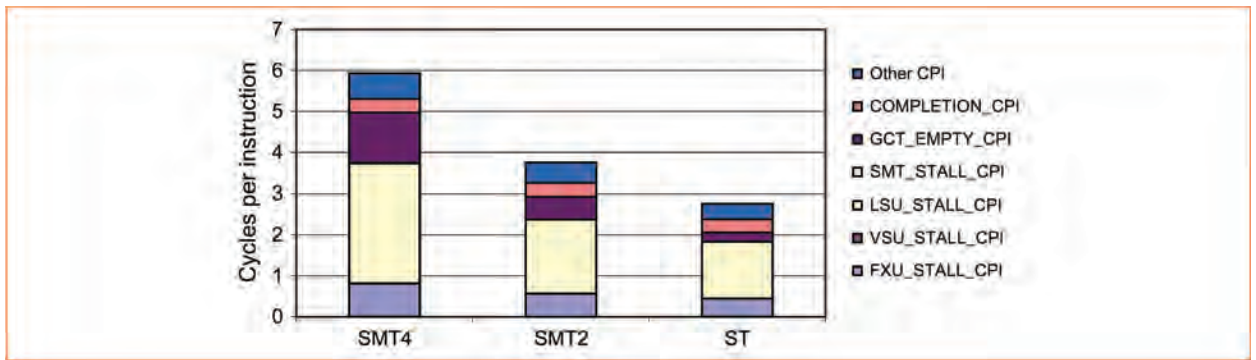


Figure 7
OLTP workload CPI components (per thread).

miss and marked cache reload and harvest the following information in software accessible registers:

- Exact instruction effective address (SIAR).
- Exact data effective address (SDAR).
- Completion stall indication and reason (MMCRA).
- Latency (MMCRA).

Software-based tools can postprocess the data from these registers over many samples and deduce the following types of information.

1. Identify the exact instruction and data address of the load miss.
2. Record the latency of load miss along with the data source of the load miss.
3. Identify load misses that were exposed to completion (load misses whose latency could not be hidden by the out-of-order speculative execution of the processor).
4. Identify load misses that take more than the expected latency from a data source. As an example, if load misses that resolve from the L2 take much more than the expected number of cycles, this is an indication of address conflicts.
5. Identify load misses that resolve from any distant or remote memory or cache.
6. Identify load misses that also experienced a translation miss.
7. Identify load misses that also suffered rejects in the load store.

Performance evaluation

Mixed ST, SMT2, and SMT4

The POWER7 core supports dynamic SMT modes that can automatically switch between the ST mode, SMT2, and SMT4, depending on workload conditions. In general, workloads with higher thread-level parallelism are obvious candidates to exploit SMT enhancements. Applications that

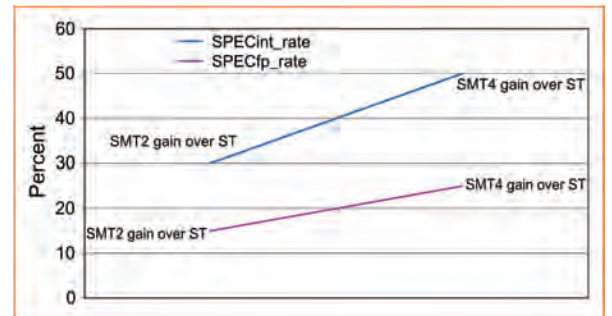


Figure 8
SMT gains versus ST.

generate asynchronous work requests, such as databases, are particularly well suited for this technology. Computationally intensive workloads, such as image and multimedia processing applications, can also benefit from SMT exploitation when library support is available.

The SAP SD benchmark, which represents the enterprise resource planning (ERP) sector, achieved a performance gain of more than $1.7\times$ in the SMT4 mode, compared with the ST mode [4.7 gigainstruction per second (GIPS) versus 2.7 GIPS], and a $1.4\times$ performance gain in SMT2 over the ST mode (3.8 GIPS versus 2.7 GIPS). GIPS is higher in multiple-threaded modes since more concurrent execution of instructions are possible (instructions in the other threads can be dispatched to go through the pipelines, even if the current thread was stalled).

On the other hand, the ST mode has the best single-threaded performance. Single-thread performance in ST mode is $1.42\times$ better than in SMT2 mode and $2.3\times$ better than in SMT4 mode. **Figure 6** shows the CPI breakdown for SAP SD. The bars represent the per-thread CPI for the workload executing under each SMT mode. A lower CPI is preferred,

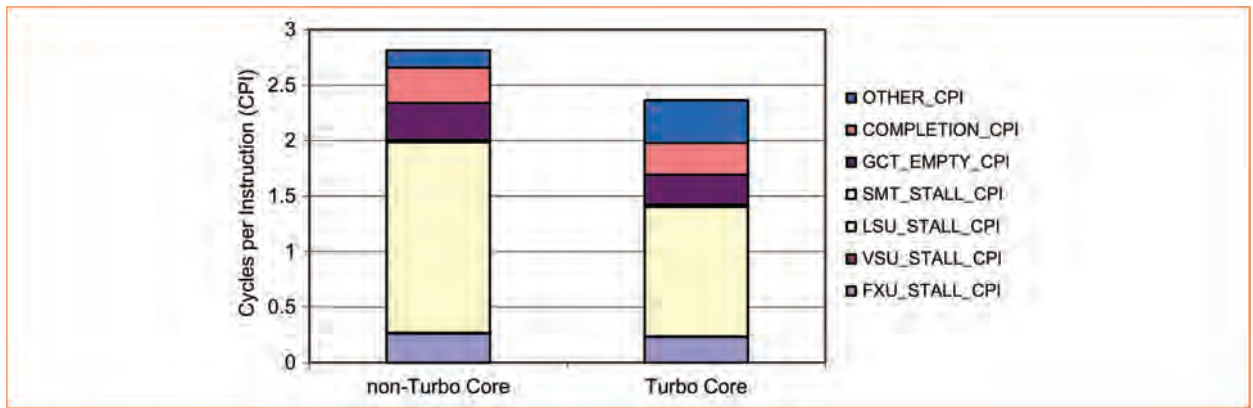


Figure 9

SPECjbb comparison of Turbo Core versus non-Turbo Core (SMT4 mode).

and the chart demonstrates that with increasing numbers of threads, the LSU_STALL component grows, as does the penalty for GCT_EMPTY. These two statistics are sensitive to the numbers of threads because each thread places additional pressure on the cache. More contention on the cache means that the threads are more likely to stall waiting for references to be fulfilled from main memory (LSU_STALL). Likewise, the global completion table (GCT) will experience stalls as more threads now wait for the next instruction.

With multiple running threads, however, the aggregated throughput capability of SMT2 and SMT4 would outperform the ST mode, although the ST mode execution may have much better ST capability.

For OLTP workloads, the SMT4 performance is almost double the ST mode performance (2.4 GIPS versus 1.3 GIPS), and SMT2 is almost 1.5× better than the ST performance (1.9 GIPS versus 1.3 GIPS). However, OLTP ST performance in the ST mode on POWER7 is 2.15× over the SMT4 mode and 1.36× better than SMT2 (see Figure 7). Again, the CPI components associated the LSU_STALL and GCT_EMPTY increase as the number of threads grows.

Throughput workloads benefit from the availability of additional threads. Figure 8 demonstrates the gains for integer and floating-point intensive workloads when running with additional SMT threads, over the ST case.

Turbo Core versus non-Turbo Core mode

Typically, an application can gain at least 10% performance in Turbo Core mode above the non-Turbo Core mode, depending on its workload characteristics and its memory footprint. There are four aspects for improved performance: the processor frequency boost, the increased L3 size, increased memory bandwidth per core, and increased memory capacity per core. Depending on how sensitive a workload is to each of these elements, the improvement seen in Turbo Core mode can

range from 8% for workloads benefiting from frequency alone, 15% for workloads that can take advantage of frequency and increased L3, and 20% for workloads that can take advantage of all four aspects, such as certain OLTP workloads. In the following discussion, *remote L3* refers to an L3 that is remote to the core but on the same chip.

SPECjbb is one of the applications that can realize significant performance gains from running in Turbo Core mode. When running in Turbo Core mode, the CPI improved by approximately 19%. The most significant improvement came from reduced LSU stall times, which is approximately 47% (see Figure 9).

The LSU stall reduction mainly came from accessing remote L3, which is much more efficient than having to access the distant memory. In non-Turbo Core mode, L1 D-cache misses have to be reloaded from memory approximately 21% of the time. L1 miss reloads from memory were significantly reduced, to only approximately 10%, whereas approximately 11% of miss reloads came from remote L3 (see Figure 10).

In addition, a 24% reduction of GCT empty cycles contributed to improved SPECjbb performance in Turbo Core mode. The lower L1 I-cache miss rate contributed to the reduced GCT empty cycles. (Note that CPU utilization is lower, at about only 90%, in non-Turbo Core mode).

The OLTP workload, which has a larger footprint compared with other workloads, was able to gain about 13% in its CPI performance (see Figure 11).

Again, the reduced LSU stall (by about 20%) contributed to the superior performance of Turbo Core mode. In Turbo Core mode, the L1 D-cache reload was satisfied by remote L3, rather than from distant memory in many cases (see Figure 12). Performance improvements also came from reduced FXU stalls (reduced by 10%) and GCT empty cycles (reduced by 9%).

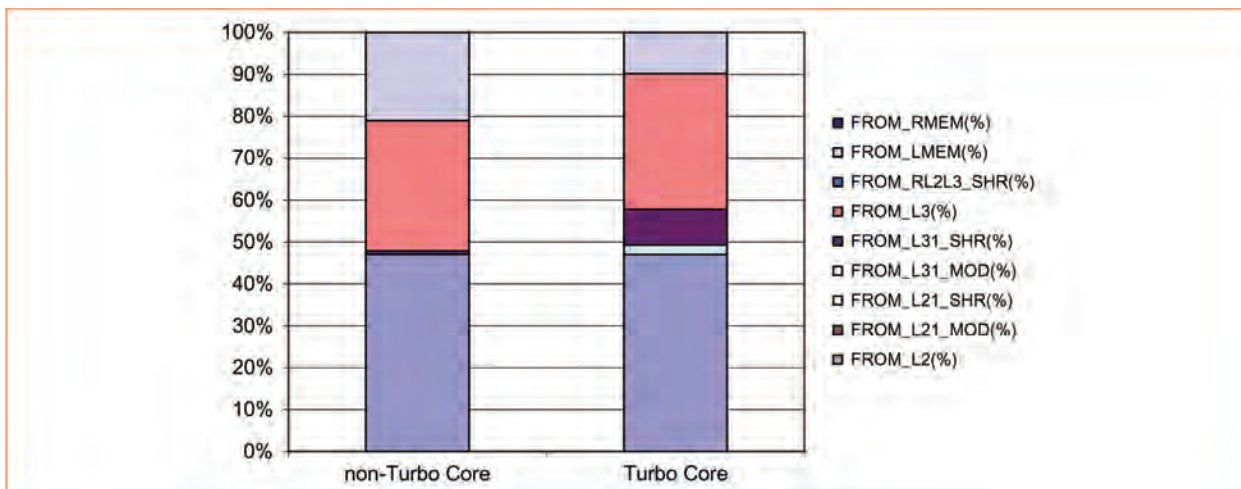


Figure 10

SPECjbb L1 D-cache miss reloads (SMT4 mode).

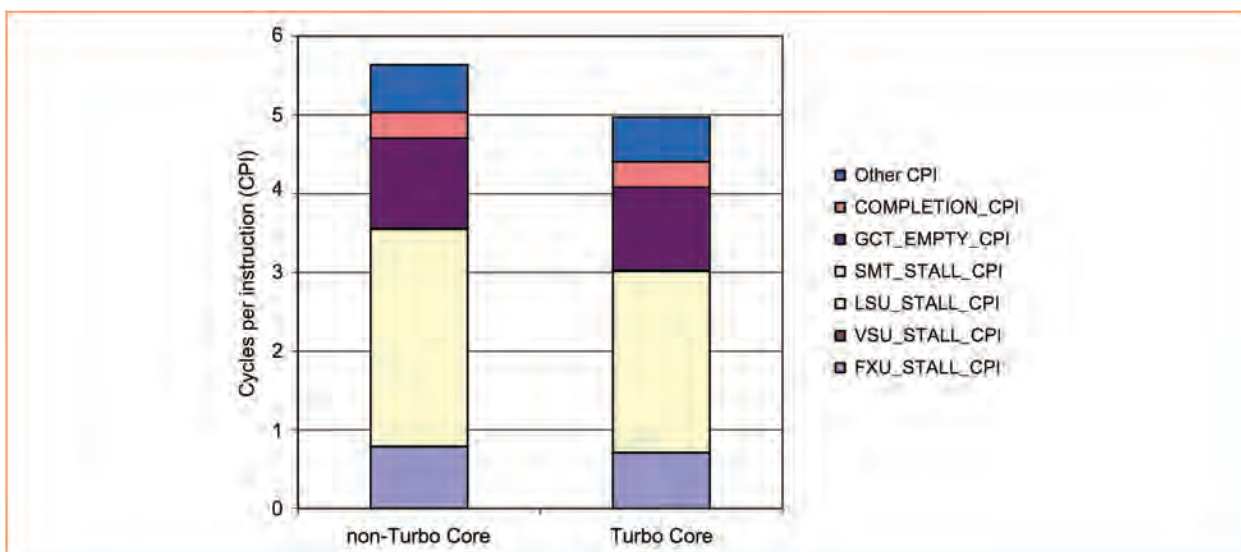


Figure 11

OLTP workload comparison of Turbo versus non-Turbo Core (SMT4 mode).

As mentioned earlier, workloads such as OLTP benefit from Turbo Core due to increased core frequency, doubled L3 capacity per core, doubled memory capacity per core, and doubled memory bandwidth per core.

VSX advantage

Linpack can take full advantage of the POWER7 processor's four floating-point pipelines and enhanced load/store capability to achieve almost 2× performance capability over

its predecessor POWER6 at higher frequency, with 4× fewer chips (four chips in POWER7 versus 16 chips in POWER6). With all cores running in the ST mode, the 16-core POWER5* at 1.9 GHz performs at 110 GFLOPs, the 32-core POWER6 at 4.7 GHz performs at 480 GFLOPs, and the 32-core POWER7 at 3.3 GHz performs at 760 GFLOPs.

Fast Fourier transforms (FFTs) also perform very well with the POWER7 processor's enhanced single-instruction multiple-data support in its VSX unit. For single-precision

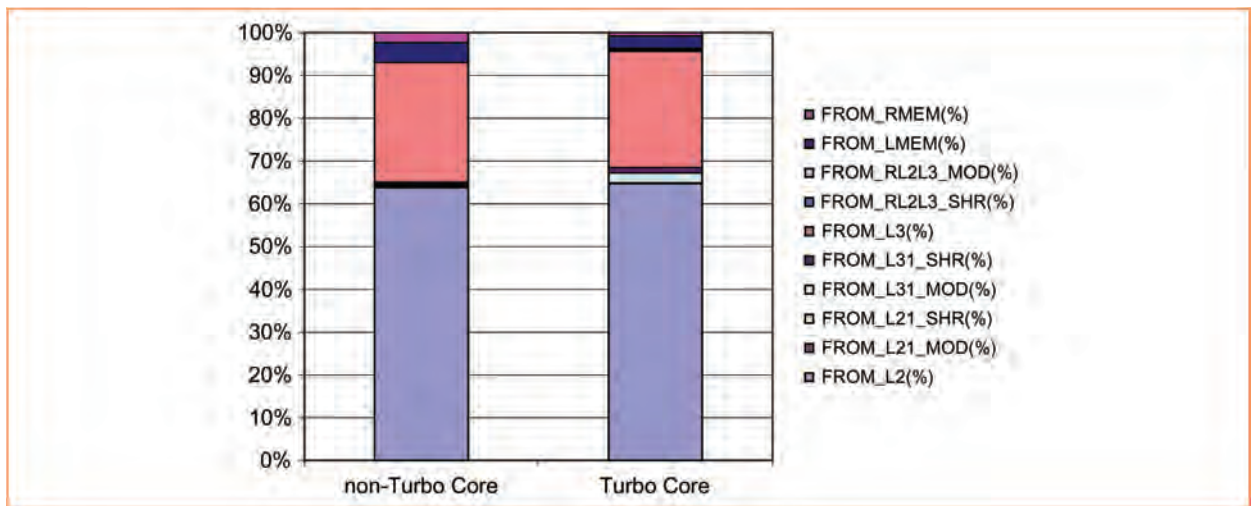


Figure 12

OLTP workload L1 D-cache miss reloads (SMT4 mode).

1-D complex FFT, POWER7 can outperform POWER5 by up to 4.95 $\times$, depending on FFT size (see Figure 13).

For double-precision 1-D complex FFT, POWER7 outperforms POWER5 by up to 2.2 $\times$, depending on FFT size (see Figure 14).

Scalability

With increased number of threads and on-chip L3, the POWER7 chip is able to provide excellent scalability. The scaling factor demonstrates how well the hardware is able to deliver improved performance with increased number of cores. The SPECjbb workload scaling results for POWER7 demonstrate near perfect scaling (see Figure 15). The baseline-scaling factor is 1 at one core, and linearly increases, indicating that with increased numbers of cores, POWER7 performance shows near-perfect scalability.

The SAP workload also demonstrates near perfect scalability for higher numbers of cores (see Figure 16).

Enhanced affinity

User-level affinity enhancements provide a double-digit performance improvement for some aspects of OLTP workloads. The affinity enhancement results on a 64-core 795 system [6] with all cores in the SMT4 mode show that the raw throughput of the OLTP workload improved by 6%, whereas the scaled transactions per minute (TPM, which is obtained by dividing the raw throughput by the CPU utilization), taking into consideration CPU utilization, gained an even more impressive 11% improvement. The CPI is reduced by 11% due to fewer stalls, less fabric traffic, and less latency.

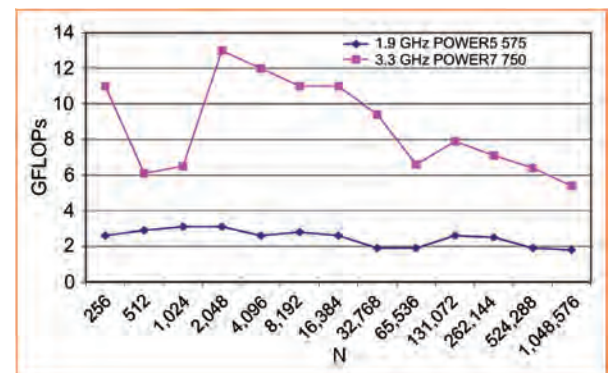


Figure 13

FFT single-precision performance on POWER5 and POWER7 (single core in the ST mode). N is the size of the Fourier transform.

Dispatch affinity

We compared the performance of a partition with dedicated POWER7 processors to that of an equivalent partition running within a shared processing pool. Both partitions ran the DayTrader workload, which is a stock trading simulator that utilizes a database, a webserver, Java, and initiates both network and disk I/O traffic. The partitions were intentionally spread across multiple processor cards to ensure that the DayTrader workload would run on various processing cores. Because of the emphasis placed on affinity for the POWER7 architecture, the SPLPAR overhead was very minimal (<2%).

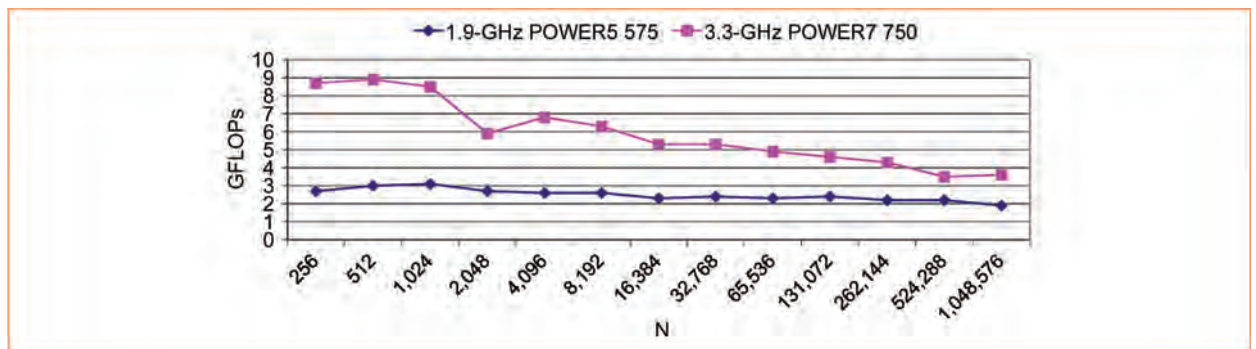


Figure 14

FFT double-precision performance on POWER5 and POWER7 (single core in the ST mode).

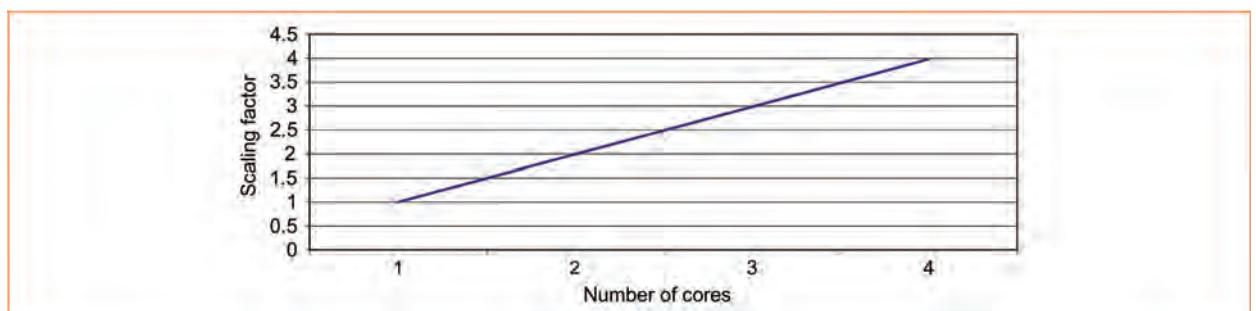


Figure 15

SPECjbb scaling on POWER7 (all cores in the SMT4 mode).

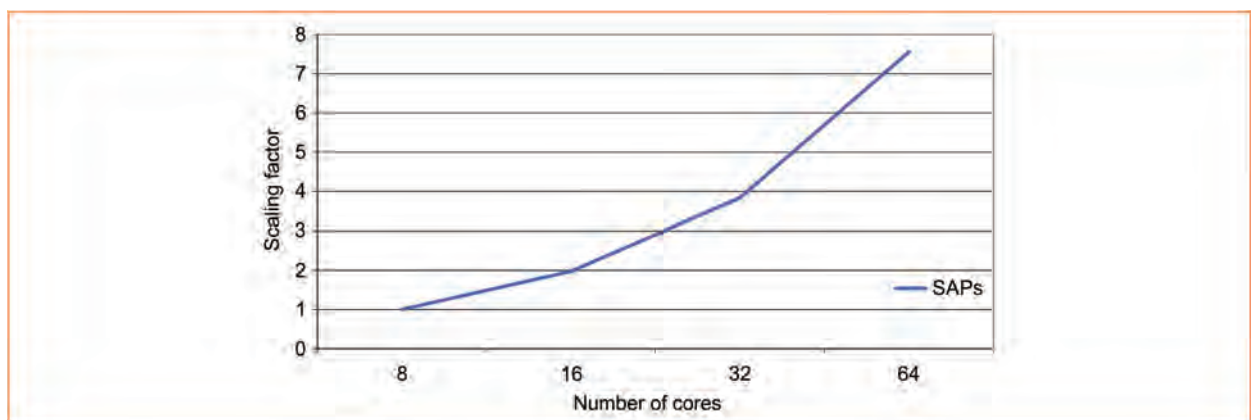


Figure 16

SAP scalability (all cores in the SMT4 mode).

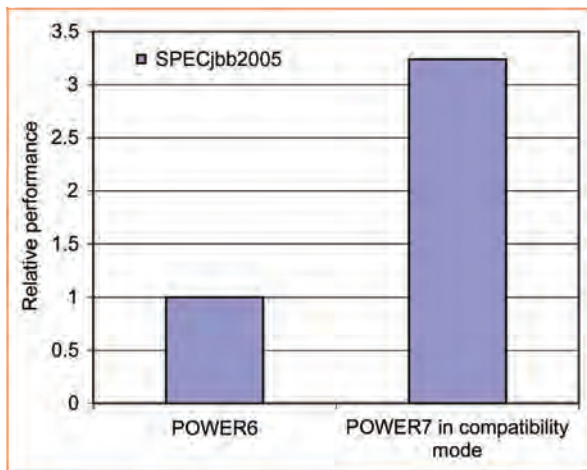


Figure 17

SPECjbb2005 in POWER6 compatibility mode of POWER7.

POWER6 compatibility mode

The results in **Figure 17** demonstrate that the same partition, migrated from an optimal environment on a POWER6 system, will perform at least as well on a POWER7 machine set to POWER6 compatibility mode. The figure compares single-socket performance where all cores are in the SMT2 mode, and POWER7 is running in POWER6-compatible mode. It is important to note that this data was for a partition, which was resident on a single chip to avoid interference of any affinity features.

Conclusion

Every aspect of the POWER7 product, from the versatile and highly scalable chip to the integrated hardware and software features, makes the smarter system technology applicable to a wide range of industry workloads.

POWER7 can handle the traditional OLTP, ERP, and database systems, as well as the new and emerging business analytics, business intelligence, and data warehousing, which were not discussed in detail in this paper. Throughput applications also benefit from the enhanced design of the POWER7 processor. The POWER7 offerings are designed for massive scale, consolidation, and parallelization capabilities, while reducing the resources necessary for management, operation, and cost of ownership, as well as energy requirements. The highly optimized out-of-order core, the on-chip cache hierarchy, the enhanced I/O bandwidth, and the new Turbo Core cache strategy provide increased performance for most workloads, and the additional affinity enhancements for user data and partition dispatching take these workloads to a new level of performance.

Eight POWER7 cores, versus two POWER6 cores per chip, provide a 4× improved compute density, while

preserving the excellent single-core performance of the previous generation, despite a lower core frequency. The performance achievements at the thread, the core, and the socket level, in conjunction with superior AIX scalability, translate to world-class system-level performance.

Acknowledgments

The authors would like to thank the following people for their contribution to POWER7 performance activities: J. Stuecheli, M. Valluri, Q. Gao, A. MacKay, D. DeSota, H. Hua, F. O'Connell, B. Konigsburg, M. D. Brown, J. LeBlanc, P. Tu, P. Vitale, D. Logan, J. Griswell, R. Bell, and A. Sawdey.

*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 Linus Torvalds, The Open Group, Sun Microsystems, Inc., or Sony Computer Entertainment Corporation in the United States, other countries, or both.

References

1. B. Sinharoy, R. Kalla, W. J. Starke, H. Q. Le, R. Cargnoni, J. A. Van Norstrand, B. J. Ronchetti, J. Stuecheli, J. Leenstra, G. L. Guthrie, D. Q. Nguyen, B. Blaner, C. F. Marino, E. Retter, and P. Williams, "IBM POWER7 multicore server processor," *IBM J. Res. & Dev.*, vol. 55, no. 3, pp. 1:1 1:29, 2011, Paper 1.
2. J. M. Borkenhagen, R. J. Eickemeyer, R. N. Kalla, and S. R. Kunkel, "A multithreaded PowerPC processor for commercial servers," *IBM J. Res. & Dev.*, vol. 44, no. 6, pp. 885 898, Nov. 2000.
3. B. Sinharoy, R. N. Kalla, J. M. Tendler, R. J. Eickemeyer, and J. B. Joyner, "POWER5 system microarchitecture," *IBM J. Res. & Dev.*, vol. 49, no. 4/5, pp. 505 521, Jul. 2005.
4. D. Guniguntala, P. E. McKenney, J. Triplett, and J. Walpole, "The read-copy-update mechanism for supporting real-time applications on shared-memory multiprocessor systems with Linux," *IBM Syst. J.*, vol. 47, no. 2, pp. 221 236, 2008.
5. S. R. Kunkel, R. J. Eickemeyer, M. H. Lipasti, T. J. Mullins, B. O'Kafka, H. Rosenberg, S. P. VanderWiel, P. L. Vitale, and L. D. Whitley, "A performance methodology for commercial servers," *IBM J. Res. & Dev.*, vol. 44, no. 6, pp. 851 871, Nov. 2000.
6. R. X. Arroyo, R. J. Harrington, S. P. Hartman, and T. Nguyen, "IBM POWER7 systems," *IBM J. Res. & Dev.*, vol. 55, no. 3, pp. 2:1 2:13, 2011, Paper 2.

Received August 25, 2010; accepted for publication April 5, 2011

Mysore Srinivas IBM Systems and Technology Group, Austin, TX 78758 USA (mysore@us.ibm.com). Mr. Srinivas is a post-graduate of Indian Institute of Technology, Bombay, an IBM Senior Technical Staff Member, and the POWER7 system performance lead. He has worked on performance-related aspects of several system components such as POWER processor, PHYP, and AIX in his 12 years at IBM and holds numerous patents across several of these areas.

Balaram Sinharoy IBM Systems and Technology Group, Poughkeepsie, NY 12601 USA (balaram@us.ibm.com). Dr. Sinharoy is an IBM Distinguished Engineer and the Chief Architect of the IBM POWER7 processor. Prior to working on POWER7, he was the Chief Architect for the POWER5 processor. Dr. Sinharoy has published

numerous articles and received more than 50 patents in the area of computer architecture, with many more patents pending. He also received several IBM corporate awards for his work in different generations of POWER microprocessors. Dr. Sinharoy is an IBM Master Inventor and an IEEE Fellow.

Richard J. Eickemeyer *IBM Systems and Technology Group, Rochester, MN 55901 USA (eick@us.ibm.com).* Dr. Eickemeyer received the B.S. degree in electrical engineering from Purdue University and the M.S. and Ph.D. degrees from the University of Illinois at Urbana Champaign. He is currently a Senior Technical Staff Member at IBM Corporation in Rochester, Minnesota, where he is the processor core performance team leader for IBM POWER servers and is working on a future processor design. Previously, he has worked on several different processor designs. His research interests are computer architecture and performance analysis. He has authored many papers and has been awarded 40 U.S. patents with others pending. He has been named an IBM Master Inventor. He has received several IBM awards including two IBM Corporate Awards.

Ram Raghavan *IBM Systems and Technology Group, Austin, TX 78758 USA (ramr@us.ibm.com).* Dr. Raghavan received a B.S. degree in physics from the University of Madras in 1979, a B.S. degree in electrical engineering from the Indian Institute of Science in 1982, an M.S. degree in computer engineering from the Michigan Technological University in 1985, and a Ph.D. degree in computer science and engineering from the University of Michigan in 1991. He is a senior engineer in the Systems and Technology Group and currently leads the nest performance modeling team that is engaged with the design team from concept to the product announcement. Prior to this role, he was part of the performance team for the Cell Broadband Engine** processor used in the Sony PlayStation** 3 game console. Dr. Raghavan joined IBM in Poughkeepsie in 1991 and moved to Austin in 1992 to take part in the development of the early verification tools for the POWER multiprocessor systems. He was the recipient of the Best Student Paper Award in Supercomputing 1990. Dr. Raghavan has written several technical papers and holds several patents.

Steve Kunkel *IBM Systems and Technology Group, Rochester, MN 55901 USA (srkunkel@us.ibm.com).* Dr. Kunkel received his Ph.D. degree from the University of Wisconsin, Madison, in 1987. He then joined IBM in Endicott, New York, doing performance analysis of a vector facility for a midrange 390 product. In 1989, he transferred to the IBM Rochester site where he currently works. During most of his years in Rochester, he has done architecture and performance analysis for AS/400* and RS/6000* (now called POWER systems) products. This included such topics as NUMA, VLIW, caches, MP cache coherency, multithreading, MP interconnects, chip multiprocessors, and converting AS/400 to PowerPC architecture processors. Currently, he is Senior Technical Staff Member and team lead and continues to work on architecture and performance analysis for POWER systems servers.

Thomas Chen *IBM Systems and Technology Group, Austin, TX 78758 USA (wtchen@us.ibm.com).* Dr. Chen received a Ph.D. degree in computer engineering from the State University of New York at Buffalo in 1989. He is a Senior Technical Staff Member responsible for systems performance in the IBM Systems and Technology Group. His technical interests include processor microarchitecture design, I/O and networking subsystem design, workload analysis and characterization, and performance modeling and analysis of processor and system architecture and designs. Dr. Chen has been awarded more than 20 patents spanning various technical areas.

W. Maron *IBM Systems and Technology Group, Austin, TX 78758 USA (maron@us.ibm.com).* Mr. Maron is an IBM Distinguished Engineer and the Chief Technology Officer, IBM Systems Technology Group for Systems Performance. He has worked at IBM for 28 years

and has achieved the level of IBM Master Inventor. After graduating from the University of Texas with a degree in computer science, he joined the IBM Federal Systems Division in Houston working on the Space Shuttle project, where he eventually led the Center of Competence for both Database and Performance for the IBM Federal Systems Division. In 1993 he joined the AIX performance team, became the team lead, and eventually led all System p* performance. In 2004 Mr. Maron was appointed Chief Technology Officer for Systems and Technology Group Systems Performance with responsibilities for System p, System i*, and System z* systems and storage. His current challenge is to ensure that the system (hardware, firmware, operating system, middleware, applications, storage) work together leveraging all IBM technical skills in order to provide the best possible balanced systems-level performance by utilizing the latest tools and driving the results of the characterization data into future products for both traditional and emerging workloads.

Diane Flemming *IBM Systems and Technology Group, Austin, TX 78758 USA (dianefl@us.ibm.com).* Mrs. Flemming holds an M.S.E. degree from the University of Texas at Austin and is an IBM Austin Site Master Inventor. She is a performance analyst in the System and Technology Group Cross-Platform Systems Performance team, responsible for software performance architecture design guidance on emerging POWER platforms. She has spent her 17-year career at IBM working on all aspects of performance, from atomic operations to full system and software stack optimizations.

Anton Blanchard *IBM Linux Technology Center OzLabs, Barton ACT 2600, Australia (anton@au1.ibm.com).* Mr. Blanchard received a B.E. degree with first-class honors from the University of Technology, Sydney. He is a Senior Technical Staff Member in the IBM Linux Technology Center. His focus is on Linux on the POWER architecture and is currently leading the Linux component of the IBM PERCS petascale high-performance computing project.

Pattabi Seshadri *IBM Systems and Technology Group, Austin, TX 78758 USA.* Mr. Seshadri joined IBM in 2002 and primarily focused on performance analysis and modeling for POWER architectures.

Jeffrey W. Kellington *IBM Systems and Technology Group, Austin, TX 78758 USA (jwkellin@us.ibm.com).* Mr. Kellington received a B.S. degree in computer engineering from the University of Illinois. He is a staff functional verification engineer for the POWER microprocessor design team. His work includes running Linux and assembly-based functional exercisers and benchmark programs on pre-silicon hardware models.

Alex Mericas *IBM Systems and Technology Group, Austin, TX 78758 USA (mericas@us.ibm.com).* Mr. Mericas is a Senior Technical Staff Member and is currently the Performance Architect for future systems. He designed the performance monitoring unit on POWER4*, POWER5, POWER6, POWER7, and PowerPC 970. He is an IBM Master Inventor with 51 patent applications and 19 issued patents covering microprocessor design and hardware performance monitors.

Andrew E. Petruski *IBM Systems and Technology Group, Austin, TX 78758 USA (petruski@us.ibm.com).* Mr. Petruski received a B.S. degree in computer and systems engineering from Rensselaer Polytechnic Institute. He is a Senior Engineer and currently the lead for chip-and system-level performance verification on POWER processors. He has held a variety of positions in microprocessor verification over his 25-year career, including functional verification on PowerPC 750* and the Cell Broadband Engine microprocessor core.

Venkat R. Indukuru *IBM Systems and Technology Group, Austin, TX 78758 USA (indukuru@us.ibm.com).* Mr. Indukuru received an M.S. degree in computer engineering from University of South

Carolina. He is an Advisory Engineer for IBM Systems and Technology Group. He currently leads the design, implementation, and verification of hardware performance monitoring infrastructure across the chip for future POWER processors.

Sergio Reyes *IBM Systems and Technology Group, Austin, TX 78758 USA (sergio1@us.ibm.com).* Mr. Reyes graduated from the University of Texas Pan America with a degree in computer science. He is an IBM Senior Engineer who has worked in processor performance for 11 years. He has focused on several areas including AIX operating system performance, System p disk I/O storage, subsystem performance, and PowerVM* virtualization. His latest initiative is to ensure that the PowerVM Virtual I/O Server maintains high performance standards as it continues to evolve.

45-nm silicon-on-insulator CMOS technology integrating embedded DRAM for high-performance server and ASIC applications

S. S. Iyer
G. Freeman
C. Brodsky
A. I. Chou
D. Corliss
S. H. Jain
N. Lustig
V. McGahay
S. Narasimha
J. Norum
K. A. Nummy
P. Parries
S. Sankaran
C. D. Sheraw
S. R. Varanasi
G. Wang
M. E. Weybright
X. Yu
E. Grabbie
P. Agnello

The 45-nm technology, called 12S and developed for IBM POWER7®, is an extremely robust and versatile technology platform that allows for a rich set of features that include embedded dynamic random access memory (DRAM), performance and dense static RAM (SRAM), a trench-based decoupling capacitor, a comprehensive device menu, and a high-performance hierarchical back-end interconnect scheme, all built on a silicon-on-insulator (SOI) substrate. Embedded DRAM was implemented for production in high-performance SOI for the first time and allowed us to leapfrog two generations of conventional SRAM densities. Immersion lithography was also employed for the first time in 45-nm IBM products. Our 45-nm design point represents a judicious leverage of silicon oxynitride dielectrics, scaled device technology, and rich features to yield chip-level performance enhancement of more than 50%, compared with our 65-nm node at comparable or less power. This paper describes the salient features of this technology node, the process architecture, the device design rationale, and the process design interactions.

Introduction

The scaling of high-performance silicon complementary metal-oxide semiconductor technology is as much an art form as a science. While scaling relentlessly proceeded until the 130-nm generation, following more or less the guidelines set by Dennard et al. in their classical constant electrical field formulation in 1974 [1], at the 90-nm generation, both gate and subthreshold leakages conspired to break classical scaling as we knew it. Strain engineering, where enhanced channel-carrier mobility was realized through uniaxial stress [along the channel: tensile for n-channel field-effect transistors (n-FETs) and compressive for p-channel FETs (p-FETs)] applied through stressor films, came to provide the required performance in place of the classical scaling.

What this meant was that the gate dielectric, and to first order the channel length, did not scale, and shorter channel transit times were mainly obtained through higher low-field mobility, as well as marginally higher saturated velocities in the high-field regions. This approach continued to the 65-nm node with the application of higher uniaxial stress.

The 45-nm node presented some interesting engineering tradeoffs for process architecture and device design, as well as feature choices, which are the subject of this paper. Lithography was clearly one of the significant challenges, and we utilized a hybrid approach with 193-nm immersion lithography for critical layers and dry lithography for the others. This is discussed in detail later. Another important decision was related to the choice of gate dielectric. Because of its low capacitance, and taking into account the ability to enhance FET performance by state-of-the-art stressors, we anticipated and have found, by carefully examining the

Digital Object Identifier: 10.1147/JRD.2011.2108112

©Copyright 2011 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied by any means or distributed royalty free without further permission by computer-based and other information-service systems. Permission to republish any other portion of this paper must be obtained from the Editor.

0018-8646/11/\$5.00 © 2011 IBM

alternatives, that the FET based on SiON (silicon oxynitride) dielectric at the 45-nm node delivers the best power/performance available. Other high-performance processes, based on high dielectric constant dielectrics, had not significantly delivered lower gate dimensions and thus delivered a much higher capacitance device, with consequently higher active power without higher frequency capability. Some very innovative structural engineering was used to lower both capacitive and resistive parasitics that could otherwise significantly detract from performance at 45 nm. The technology was optimized for system-on-chip (SOC) applications and provides a judicious set of threshold-only tunable device operating points that allow a single technology definition to span a leakage range of 4.5 orders of magnitude, allowing for the simultaneous manufacture of the industry's highest performing server and mainframe designs in addition to a well-optimized general-purpose application-specific integrated circuit (ASIC) technology for SOC applications.

A key distinguishing feature of our 45-nm technology, which we call 12S, is the special attention paid to embedded memory. In addition to an extensive menu of embedded static random access memory (eSRAM) cells ranging in size from 0.462 to 0.315 μm^2 and spanning a corresponding range of performance, our 45-nm technology boasts the first silicon-on-insulator (SOI)-based embedded dynamic RAM (eDRAM) technology with a cell size of 0.067 μm^2 . Most die today are dominated by memory, and it is not uncommon for high-end processors to have as much as 70% of the die area allocated to the memory. Our trench-based eDRAM provides a 3.3 times area advantage at the application level at a mere 12% cost adder at the wafer level. This means that when approximately 10% of the chip area is contiguous SRAM (as in a cache), the reduction in die size more than pays for the extra processing cost for eDRAM. Additionally, the ability to use the trench capacitors for midfrequency noise decoupling translates to an additional performance boost at the chip level of approximately 4%–7%.

Taken together, the holistic embedded memory approach that we have taken allows us to leapfrog memory density by almost two generations. Equally important to note is that the power allocated to memory was reduced by more than three times, a key consideration in high-performance processors. Finally, trench-based eDRAM has a more than 1,000 times reduction in soft-error rates, which ensures what is perhaps the most reliable 45-nm technology.

The IBM 45-nm SOI technology is the first comprehensive technology that completely and effectively addresses the server and the SOC space. The only segment that the 45-nm SOI technology does not address is the ultralow-power portable electronics market. While a fully depleted SOI technology can address this ultralow-leakage segment, we point out that the IBM 45-nm SOI technology is partially depleted. To address the ultralow-leakage segment,

IBM has developed a bulk technology with 100 channels that has been described elsewhere [2].

In this paper, we discuss the details of the technology and key results. We start with immersion lithography and then discuss the front-end [including the deep trench (DT) process] and interconnect architectures, followed by a device design including the eDRAM pass transistor. We also discuss the passive elements offered in this technology. We limit our discussion to the process and parametric results. Functional results that depend on circuit design and architecture are dealt with in other papers in this issue.

Immersion lithography

Lithographic resolution scaling has historically been governed by periodic reductions in the wavelength of imaging radiation λ while increasing the maximum available numerical aperture (NA), i.e.,

$$\text{resolution} = \frac{k \times \lambda}{\text{NA}}$$

$$\text{NA} = n \times \sin \theta, \quad (1)$$

where k is the process factor, n is the index of refraction in an imaging medium, and θ is the maximum incident angle of imaging radiation normal to wafer surface. As the 45-nm technology node approached, wavelength scaling efforts directed toward moving from 193- to 157-nm imaging radiation were abandoned across the semiconductor industry because of insurmountable technology obstacles, including insufficient supply of optically acceptable CaF_2 lens material and a lack of protective photomask pellicles and photoresists with optical densities suitable for use with 157-nm radiation. This left a critical void in lithographic technology roadmaps [3]. In the next step past 157 nm, extreme ultraviolet (UV) technology (13.4 nm) remained years away from manufacturing and, even today, has a questionable future [4]. At the same time, NA scaling via improved lens designs was rapidly approaching a theoretical limit of 1 (limited by index of refraction of air at 1.0) with production tool roadmaps ending at 0.93 NA [5].

This realization led to two predominant pathways in the industry. The first was to impose extreme design restrictions to enable extraction of additional life out of existing 193-nm lithography scanners operating in the less than 1-NA regime, including implementation of double-exposure methodologies. This approach can be generalized as placing disruptive challenges on the design teams for 45 nm while allowing more time for advanced process development to prepare for 32 nm. Although this approach has proven to be workable for large-volume and low part number manufacturers [6], this approach was far less suitable for lower volume designs found in most diversified integrated device manufacturers and foundries where it is

crucial to enable aggressive on-schedule migration of intellectual property.

The alternative pursued by IBM was to break the theoretical NA limit of 1 by placing the entire wafer imaging process under water. Since the theoretical NA limit, and in turn, resolution capability, is directly proportional to the index of refraction of the imaging medium, this enables a 1.44 times improvement in resolution by taking advantage of the index of refraction of water (1.44) versus air (1.00). To contrast with the aforementioned case, this approach can be generalized as placing disruptive challenges on the process development teams for 45 nm while buying design teams additional time to anticipate and migrate toward more restrictive design rules for the 32-nm node. This process-challenged approach for 45 nm was particularly well suited to leverage IBM's unique development alliance ecosystem, incorporating close collaboration with exposure tools, resist materials, and mask vendors to deliver a viable immersion manufacturing process in time for 45 nm.

Many critical challenges were overcome during immersion lithography implementation, the greatest of which was achieving defect density targets on par with traditional "dry" systems. In order to achieve cost-effective throughput, it is necessary to move the wafer at speeds up to 50 cm/s while maintaining control of a water "disk" between the lens and the wafer surface. An operation at such speeds requires precise control of water evaporation and control of the meniscus associated with the fluid disk (see **Figure 1**). Evaporation of water droplets can lead to both defects and random evaporative cooling that contributes to thermal imbalances [7]. These thermal imbalances can then lead to random uncorrectable overlay errors.

Fluid control and protection from contamination during exposure is driven by carefully tuned and optimized surface properties delivered via topcoats that cap traditional photoresist imaging stacks. Working with Japan Synthetic Rubber Corporation, which is a materials development alliance partner, IBM's fluoroacrylate technology has formed the basis of today's industry-leading topcoat material [8]. New characterization techniques such as the water extraction and analysis technique developed at IBM [9] have also been developed to monitor leaching from underlying materials through topcoats. Again, consistent with the process-challenged model, many such materials innovations have been a critical component of immersion lithography learning [10].

IBM's approach to defect learning leveraged two generations of immersion prototype tools ahead of 45-nm manufacturing, focusing on mature 90-nm IBM PowerPC* processors as defect learning vehicles. In December 2004, IBM announced that it was the first manufacturer to have demonstrated a fully functional processor that included immersion lithography processing [11] and, by 2006,

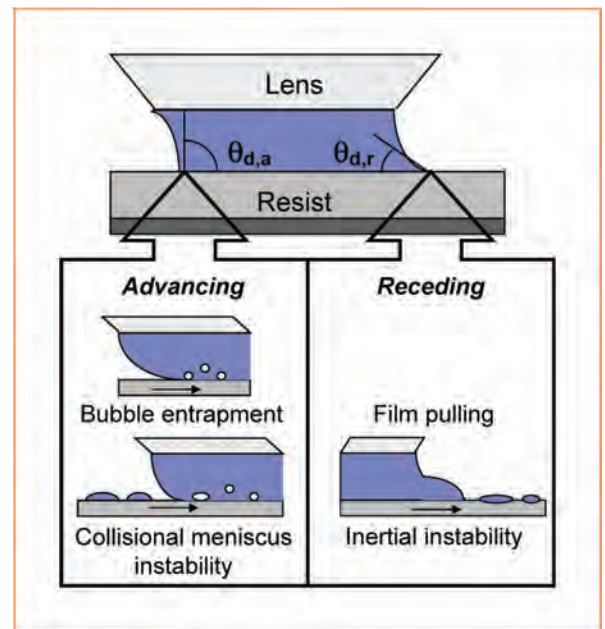


Figure 1

Schematic of mechanisms impacting immersion defectivity and scan throughput.

exceeded the International Technology Roadmap for Semiconductors for patterned wafer defectivity [12].

The thermal challenges associated with random droplet residue and evaporative cooling described earlier represent the most egregious types of thermal distortions that can lead to problems with overlay control. Beyond such cases, however, more subtle thermal variations during "perfect" wafer processing are an intrinsic part of immersion lithography and continually challenge overlay control, as each tool presents its own thermal "signature" based on variations in machine tolerances. IBM has developed and implemented fabrication control systems to continuously track, correct, and respond to variations in thermal signatures across multiple immersion tools, including the introduction of nonlinear die-by-die overlay grid corrections to compensate thermal signatures leading to overlay errors [13]. Such control systems have been proven to be essential in an era with more aggressive overlay requirements than classical scaling would dictate due to limited gate dimension scaling achieved in a FET design.

Imaging at NA much greater than 1 introduced new optical challenges never before experienced in semiconductor manufacturing. Reflectivity control was correctly anticipated as a significant challenge well in advance of 45-nm manufacturing, leading to the development of novel multilayer antireflective stacks developed in conjunction with IBM's material development partners and later

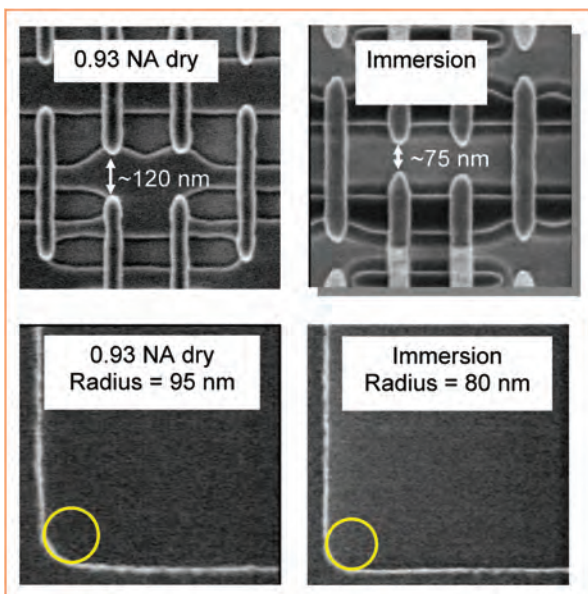


Figure 2

Improved 2-D pattern fidelity enabling SRAM scaling.

commercialized for broader industry use [14]. Polarization effects required development of new mathematical treatments to describe the optical pupil [15]. Photomasks were also challenged with new requirements such as reduced mask birefringence in order to maintain critical dimension control. Again, through IBM's process development ecosystem, new partnerships have been forged to enhance IBM's existing internal mask development capabilities [16].

The device benefits of immersion are readily observed in a typical SRAM cell, as well as fundamental test structure measurements, as evidenced in **Figure 2**. Increasing lithographic resolution enables reduced gate tip-to-tip spacing, reduced active-to-active spacing, and improved corner rounding control (Figure 2). This imaging capability may be exploited to drive maximum density scaling or, alternatively, may be exploited for improved device uniformity and performance, as market demands require.

These results are achieved using a *single*-exposure pass and thus are indicative of the fundamental imaging capability. Capability can be further enhanced through the use of double-exposure techniques that enable even further gate end control at additional cost and design complexity.

IBM has leveraged immersion lithography throughout 45-nm 12S technology, including shallow trench isolation (STI), gate, contact, and all $1\times$ metal/via levels ($1\times$ is the minimum pitch). As noted earlier, the decision to implement immersion lithography throughout 45-nm processing was driven by a desire to maintain maximum density scaling via improved resolution with maximum design flexibility

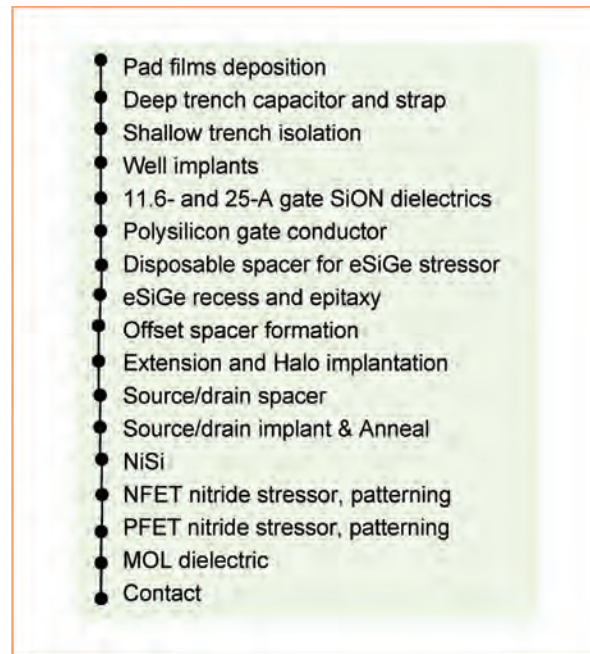


Figure 3

Process flow for 45-nm SOI technology FEOL.

and straightforward migration of intellectual property by avoidance of highly restrictive design rules. At the same time, IBM's aggressive immersion process development roadmap has built up deep expertise with a technology that is solidly entrenched as a critical piece of the lithographic roadmap for delivering 32-, 22-, and even 15-nm technology.

FEOL process

The front-end-of-line (FEOL) process delivers numerous circuit elements on the same chip to satisfy the performance and bandwidth demands of high-performance processors. It delivers not only an extremely high-performance set of logic FETs but also integrated memory of two types (both eDRAM and SRAM), as well as higher voltage (1.5–1.8 V) FETs and passive components for analog and input/output (I/O) operation. As illustrated in **Figure 3**, the flow begins with eDRAM storage cell DT capacitor formation, continues through FET and performance stressor levels, and finishes with contact formation.

eDRAM and the DT capacitor

The high-performance eDRAM design prototypes first fabricated on the IBM 65-nm SOI platform [17, 18] set the stage to bring the traditionally off-chip bulk substrate eDRAM cache onto the microprocessor chip for the 45-nm SOI technology generation. The DT capacitor storage node

Table 1 Device offering and typical parameter values at 1.0 V 25 C (11.6 and 16 A t_{ox}) and 1.5 V 25 C (25 A t_{ox}). All the devices shown are floating-body devices. Additional devices are offered as body contact (not shown). Values are not self-heating corrected and not necessarily reflective of ac switching current due to floating-body effects. Note I_{eff} average of I_d ($V_{GS} = V_{DD}/2, V_{DS} = V_{DD}$) and I_d ($V_{GS} = V_{DD}, V_{DS} = V_{DD}/2$). (DG: double gate.) (^aXVT device is a limited offering.)

Dielectric thickness	Name	L_{gate}	V_{dd} (V)	n-FET at 1.0 V (DG 1.5 V)			p-FET at 1.0 V (DG 1.5 V)		
				V_{tsat} (mV)	I_{off} (nA/ μm)	I_{eff} ($\mu A/\mu m$)	V_{tsat} (mV)	I_{off} (nA/ μm)	I_{eff} ($\mu A/\mu m$)
11.6 Å	LVT	40	0.9–1.1	75	2,200	825	33	1,060	575
	RVT	40		165	300	670	125	100	485
	HVT	40		205	140	570	175	35	425
	SVT	44		245	47	490	195	14	390
	UVT	44		350	4.3	315	287	2.0	265
16 Å ^a	XVT	47	1.5–1.8	480	0.93	125	430	0.29	103
25 Å	DG	90		175	30	605	210	4.9	330

not only delivers unparalleled functionality opportunity in the form of dense on-processor eDRAM memory but also offers designers the area-efficient DT capacitor to provide high-density decoupling.

Compared to the fabrication process for prior bulk substrate eDRAM technologies, the DT capacitor fabrication on SOI substrates adds some process and control challenges to the initial trench patterning sequence but in the end enables both a significant net process cost and complexity savings and a net process window improvement versus its predecessors. The vertical “collar” isolation, which must be fabricated in bulk technologies, is enabled instead in the SOI generation by the built-in buried-oxide layer in the substrate. In addition, the arsenic (As) doped glass (ASG) solid-source diffusion process sequence required for appropriate buried plate As doping levels in prior generations is replaced in the 45-nm SOI generation with a simpler ion implantation sequence. The ASG plate doping process requires thick DT sidewall spacers during wet clean to protect the buried-oxide layer and to prevent As from diffusing into the device active area. These thick spacers, which are put in place following initial etch steps, limit the trench dimension and depth during the subsequent etch steps through underlying silicon and thus limit capacitance. A high-dose ion implantation process performed at close to vertical angles into the trench opening has been developed to dope the trench-capacitor bottom plate with acceptable resistivity. It is compatible with a thinner spacer and allows for reduced process complexity and cost. Fabrication process challenges include protecting the physical integrity of the buried-oxide vertical isolation; protecting the electrical integrity of the SOI active region to be used for later access transistor fabrication; enabling the reduced scaling pitch of the 45-nm cell design, without shorting or crosstalk between neighboring cells; and defining a large enough process window to achieve consistent results across chip, across

wafer, and across all products through the technology life cycle.

DT patterning begins with a blanket film stack of pad oxide, pad nitride, a substantial mask oxide film, and 193-nm trilayer dry photolithography. Mask-open reactive ion etching (RIE) is performed through seven films including a specified overetch beyond the buried-oxide layer. A detailed pullback and protective liner deposition sequence enables the subsequent DT RIE, wet trench bottling, and ion implantation processing to define the DT capacitor and buried plate without disturbing the physical integrity of the buried-oxide vertical isolation or the electrical integrity of the SOI active region.

DT capacitor node dielectric, As-doped polysilicon, and DT-to-transistor strap formation largely follow traditional fabrication process sequencing, but in 12S technology, we have introduced enhanced in-line metrology, advanced process control modeling, and shorter term feedback. Collectively, these enable much improved control over potential process tool drifts and structural consistency across different products.

FET fabrication

Steps subsequent to DT and shallow trench formation are primarily devoted to FET formation. Outside of the memory FETs (which have fixed geometry), the technology offers a menu of general-purpose FET types, as detailed in **Table 1**. Implants prior to gate dielectric formation provide for threshold adjustments and differentiate between the logic FET threshold voltage V_T types. Two gate dielectric thicknesses (11.6 and 25 Å equivalent thickness) are used on all products and a third intermediate dielectric for low leakage (16 Å) has been qualified and may be used on selected products.

After gate formation, a series of spacers, implants, and stressors are defined. First, a disposable spacer is formed

on the gate sidewalls of p-FETs, a recess is made into the p-FET source drain regions, and embedded SiGe stressor is grown [19]. Following removal of this eSiGe disposable spacer, the device architecture utilizes an offset spacer to separate the extension and halo (so-called pocket) implants from the gate conductor, resulting in minimum drain capacitance to the gate. All thin oxide n-FETs or p-FETs share extension and halo masking and implantation. This is to minimize postgate processing and thus electrical variability. A last nitride/oxide spacer is formed to provide further separation of the source and drain implantation, as well as silicide from the gate and channel regions.

Key performance elements are listed in references [20, 21]. Logic p-FETs benefit from close proximity to the embedded SiGe stressor in the source–drain regions with deep cavity and compressive stress nitride overlayer. This combination increases the hole mobility by approximately $4\times$ over a nonstrained Si control.

Logic n-FETs benefit from techniques of stress memorization [22], tensile stress liner overlayer, and spacer proximity technique enabling closer proximity of the tensile stress liner overlayer to the channel [23]. In addition, resistance reduction is achieved through an advanced laser millisecond anneal process.

Device performance also benefits from such details as gate height reduction to minimize the parasitic capacitances associated with the gate sidewall coupling, etch control to minimize resistance increases from film removals, and carefully designed junction butting (to buried oxide) implantation to minimize subthreshold slope degradation and device punchthrough effects.

Low-leakage logic

With the 45-nm node, for the first time, the IBM ASICs business chose to leverage SOI technologies to provide the most competitive technology possible for that marketplace. A low-leakage device to satisfy the lower power set of applications most relevant to this market was developed. Thus, beyond the high-performance processor V_T types (LVT, RVT, HVT, SVT), the menu was expanded to capture the low-leakage UVT and XVT devices. With the body region of the FET floating, junction leakages made the task generally problematic because they manifested themselves as a source-to-drain leakage that could not be controlled by the gate. Containing the junction leakages within a high-performance technology that required abrupt junctions was one of the challenges faced in developing these low-leakage devices.

During the development of the low-leakage FETs, interactions between the floating body and the drain-to-body leakage were observed to create unique challenges. As the channel doping was increased, the impact ionization in the drain–body junction also increased. This increase in leakage resulted in an increase in the body potential, which in turn

reduced V_{TSAT} . Thus, the higher dosage, normally used to increase V_{TSAT} , provided a compensating effect through the body voltage modulation, eventually requiring very high channel doping to achieve the desired V_{TSAT} . The increased doping in turn resulted in excessive drain impact ionization, resulting in drain–source leakage through the body to the point where this leakage became a significant part of the overall off-current and became larger than the subthreshold leakage for the lowest leakage FETs. This posed a challenge to achieving the desired off-current target for the lowest leakage FETs. A key to breaking this cycle was to reverse scale the lowest leakage FETs by increasing the gate length. This reduced the off-current by increasing V_T as well as by improving the subthreshold slope (at a given channel doping level). The gate length for the SVT, UVT, and XVT V_T FETs were consequently pushed longer for optimum power and performance, with the lowest leakage device (XVT) established at the longest length allowed by the technology. While this resulted in some increase in dynamic power from the added gate capacitance, the improvement in static power—both in the nominal and in the worst case corner—far outweighed the increase in dynamic power. The alternating current (ac) drive current in the switching transient is significantly enhanced by the higher body potential created by the capacitive coupling of the body to the switching gate. Based on the simulations of comparable bulk technologies, the net effect is that a power/performance advantage is retained across the spectrum of high-leakage–low-leakage SOI devices. Furthermore, complex effects such as historically seen with the aforementioned coupling were scrutinized and accurately captured by the compact models and verified over a large range of circuit topologies, operating voltages, V_{DD} , and temperature ranges. Overall, the 45-nm SOI technology extends the range of high-performance FETs to the low-to-moderate power space with a well-characterized set of compact models.

eDRAM access FET

Sharing the thick-oxide dielectric at 25 Å with the I/O devices, the eDRAM access FET device presented unique challenges well beyond what may be viewed as a simple size reduction of the 25-Å I/O device. The device design was required to account for threshold variability, which is intimately tied to memory retention time. It was also required to minimize junction leakage and gate-induced diode leakage current to minimize the floating-body potential and its impact on retention. Primarily, these limitations and improvements for them are learned through functional evaluation and experimental modulations of array electrical failure signatures. Several dedicated process steps are put in place to provide the eDRAM FET the tuning required to meet performance specifications.

Device and array co-design was critical for success. The access device off state is achieved with wordline low voltage

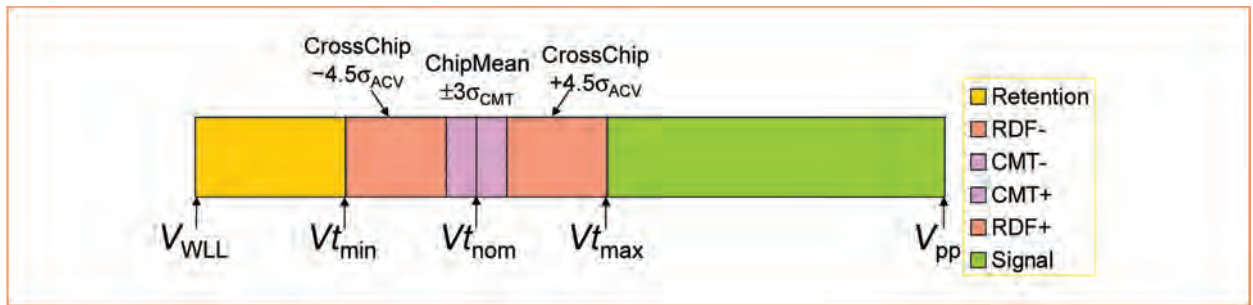


Figure 4

Schematic of wordline swing “budget” being applied to meet retention, device V_T variation and gate overdrive requirements. Typical redundancy can tolerate an average fail density on the order of a few bits per million ($\sim 4.5 \sigma$). CMT is the chip mean tolerance (i.e., chip-to-chip variation). ACV is the across-chip variation tolerance (i.e., within-chip variation).

V_{WLL} applied on the gate for data retention and turned on with gate overdriven to wordline high voltage V_{PP} by internal power supply to allow the storage capacitor to be charged to V_{DD} . Wordline swing ($V_{PP} - V_{WLL}$) is limited by the reliability of the 25-Å gate oxide. The wordline swing “budget” is spent to guarantee data retention and cell performance, as well as statistic variation of threshold voltage, as illustrated in **Figure 4**.

The access device is required to scale at a similar pace as the eDRAM cell size. At ever-smaller dimensions, controlling threshold voltage variability dominated by random dopant fluctuation (RDF) may be the most challenging task facing access transistor scaling. RDF V_T variation is strongly dependent on device footprint, doping level, N_A , and oxide thickness, t_{ox} , as highlighted Equation (2) [24]. A 60% scaling factor implies that V_T sigma increases by approximately 30% at each technology node, and if not controlled, it would consume an unacceptable portion of the wordline swing budget because of the 9-sigma requirement for array fixability. In order to control RDF-induced V_T variability, we aggressively lowered channel doping concentration and the device V_T . Consequently, we adopted negative wordline low to achieve the OFF-state of the access transistor and maintain acceptable across chip V_T variation as

$$\sigma V_T = 3.19 \times 10^{-8} \frac{t_{ox} N_A^{0.4}}{\sqrt{L_{eff} W_{eff}}} [V]. \quad (2)$$

Interconnect technology

The 45-nm node copper-interconnect scheme consists of a single damascene M1 metal layer followed by as many as 12 dual damascene wiring levels. It provides the designer with several combinations or hierarchies of “thin wires” and “fat wires” to choose from, depending on the particular application and design methodology. A cross-sectional scanning electron microscope (SEM) image of a ten-level

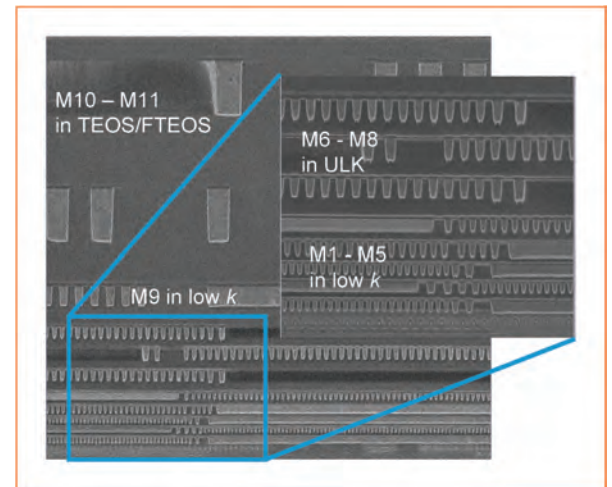


Figure 5

SEM cross-sectional image of an 11-level BEOL build. The levels are grouped by the type of dielectric in which they are built. (Inset) A “zoomed-in” view of the tighter pitch levels.

interconnect structure is shown in **Figure 5**. The thin wires can be drawn at the technology-allowed minimum pitch and are thus characterized by high density and high electrical resistance, making them suitable for local interconnects. They are built in a low dielectric constant ($k = 2.7$) film, which is a 10% lower k than that used in the 65-nm node. The fat wires are offered in a number of pitches, all of which are multiples of the minimum technology pitch. The increased wire width and spacing at the fat-wire levels leads to reduced parasitic resistance and capacitance, making them suitable for global wiring and power distribution. A new ultralow dielectric constant ($k = 2.4$) film (ULK) was introduced into a select number of fat-wire levels in

order to provide a further reduction in the parasitic capacitance and power dissipation. Since the fat wires are by definition less demanding from a lithographic and patterning point of view, they are an ideal entry point for new materials for back-end-of-line (BEOL) interconnections.

POWER7 uses an eleven-level BEOL build. The bottom three levels (M1–M3) and their associated vias (V1–V3) can be drawn at a pitch as low as 140 nm, which is a $0.7\times$ scaling of the 65-nm technology node. This allows for a $2\times$ reduction in area or double the density. To get the improved lithographic resolution and depth of focus required at these $1\times$ levels, the immersion lithography solution described in the previous section is used. Silicon-containing bottom antireflective coating was developed and implemented at the M1 level for reduced reflectivity at high NA [25]. The dual damascene levels (M2 and above) use a via-first patterning approach, whereby via lithography and RIE steps are performed first, followed by the associated line level patterning steps. This sequence allows for reduced topography and a wider depth of focus during via patterning.

While the M1–M5 levels employ dense carbon-doped SiCOH films, the M6–M8 fat wires, which are drawn at $2\times$ minimum pitch, are built in a porous SiCOH dielectric. Both types of dielectrics are deposited by plasma-enhanced chemical vapor deposition methods [26]. Porosity is introduced into the films using an organic porogen, which is later removed by an UV curing processes, leading to a reduced dielectric constant ($k = 2.4$). The resulting porosity is 25% by volume, and the average pore diameter is approximately 2 nm. It is important that the interlevel dielectric films be resistant to plasma damage, particularly during the RIE patterning processes. Damage manifests itself in reduced carbon content and a corresponding increase in the dielectric constant. Good agreement between the measured and the simulated capacitance at these $2\times$ levels is consistent with minimizing this damage. At the same time, a reduction of approximately 15% in capacitance, relative to dense SiCOH films, was observed [27]. This is particularly important for the fat-wire levels where long runs can lead to high parasitic capacitance and circuit delays. With porosity and reduced dielectric constant comes a lowering of the film mechanical strength. Extensive reliability testing including chip-to-package interaction tests have been performed as part of the technology qualification. The ULK levels passed the same reliability criteria as the more dense SiCOH films used in the thin wires [26].

Additional strength is added to the BEOL stack by incorporating high modulus materials at the M9–M11 layers. The M9 level on the POWER7 processor is built in a dense SiCOH ($k = 3.0$) dielectric and is designed in a $4\times$ minimum technology pitch. The two top levels, i.e., M10 and M11, are built in a combination of tetraethyl-orthosilicate (TEOS) oxide dielectrics and fluorine-doped TEOS oxide

dielectrics. These levels, which are drawn at a “coarse” $0.8\text{-}\mu\text{m}$ line and space, also provide the necessary low resistance for global signal and power distribution across the chip.

Passive elements

This technology offers a full suite of front-end and back-end passive devices. The front-end passive devices include a precision resistor, forward-bias ideal diode, and multiple capacitor offerings for various voltage and capacitance density needs. The precision resistor is a boron-doped polysilicon resistor formed on the STI with a nominal sheet resistance of $350\ \Omega/\text{sq}$ and a $\pm 10\%$ tolerance. The forward-bias diode is built using the P+/N-well junction from the thick-oxide p-FET. The diode layout is optimized for improved ideality, achieving a nominal ideality of 1.01 and ± 0.006 tolerance. Because the junction is defined by a polybounded annular layout, good junction ideality is achieved by avoiding contact with the STI. Additionally, by rotating the long axis of the diode 90° relative to the FET orientation, the two-way halo implant is largely eliminated from the diode perimeter, further improving the ideality. This ideal diode is suitable for various applications ranging from thermal sensing to band-gap reference circuits. There are two planar gate oxide capacitors formed in the N-well regions, i.e., one with the $12\text{-}\text{\AA}$ oxide and the other with the $25\text{-}\text{\AA}$ oxide. Both are fully characterized and modeled for high-frequency operation as a varactor up to 0.9 V ($12\ \text{\AA}$) and 1.8-V decoupling ($25\ \text{\AA}$). The thin-oxide varactor has a tuning range [C_{max} (+0.9 V) to C_{min} (–0.5 V) ratio] of 5.3 for a gate length of $0.472\ \mu\text{m}$.

A high-density DT decoupling capacitor ($\sim 270\ \text{fF}/\mu\text{m}^2$) is formed by the parallel connection of the eDRAM DT capacitors and results in a $25\times$ capacitance density advantage over the planar $25\ \text{\AA}$ decoupling capacitor. Each DT has a nominal capacitance of 18 fF and a total resistance of 12 k Ω . For the POWER7 processor, the addition of $10\times$ larger on-chip decoupling capacitance greatly reduces on-chip power supply variation and noise transients. By ensuring that the worst condition within chip voltages remains very close to the voltage applied to the package, the system can achieve either higher performance at a given voltage or lower power operation at reduced voltage for a given performance level. Because the high aspect ratio of the trench results in a distributed resistance, the effective resistance is approximately 4 k Ω , which determines the frequency response of this capacitor.

The back-end passive devices consist of an inductor and a capacitor. The inductor is constructed from spiral loops of the copper wiring levels and is supported for the $1\times$, $1.3\times$, $2\times$, $4\times$, $6\times$, and $10\times$ levels. A vertical native capacitor is formed by interdigitated metal and via comb structures in any consecutive $1\times$, $1.3\times$, or $2\times$ copper wiring levels. The capacitance of the vertical native

Table 2 SRAM bit cell offerings for IBM’s 45-nm SOI technology. For those cells using dual supply voltages, the supply voltage for the cells and wordline is higher than the bitline. For dual supply voltage cells, the smaller voltage is the minimum bitline voltage, and the higher voltage is the minimum cell and wordline voltage.

<i>Cell name (unit/definition)</i>	<i>Type</i>	<i>Cell size (μm^2)</i>	<i>Cell leakage nom 25C (nA)</i>	<i>$V_{\min}$ time 0 at circuit (V)</i>
S462	Single port	0.462	16	0.8/0.95 dual supply
P404	Single port	0.404	13	0.8/0.95 dual supply
F404	Single port	0.404	18	0.8
P315	Single port	0.315	6.5	0.8
DP608	Dual port	0.608	8	0.8
Two-port	Two port	0.638	15	0.8
Ternary content addressable memory	Ternary content addressable memory	1.065	6.5	0.8

capacitor is independent of the voltage, unlike the metal–oxide–semiconductor-based capacitors.

Memory

IBM’s 45-nm SOI SRAM cell menu satisfies a wide range of applications (see **Table 2**). Single-port six-transistor cells from $0.462 \mu\text{m}^2$ for high performance to $0.315 \mu\text{m}^2$ for high density are offered. Dual-port, two-port, and ternary-content-addressable memory cells are also offered. Each bit cell’s transistor dimensions, currents, and threshold voltages V_T are optimized to provide designers different tradeoffs of density, performance, standby current, and operating voltage. All the cells are fabricated with a subset of the strain enhancement techniques available in the technology—that is, single tensile stress liner to enable narrow n-FET to p-FET spacing and eSiGe integration for improved p-FET performance. All single-port cell layouts are “thin cell” [28]. This minimizes bitline wiring length and allows a single orientation of polysilicon gates in the cell (see **Figure 6**).

The S462 ($0.462\text{-}\mu\text{m}^2$) cell is used in high-performance small arrays such as the POWER7 Level 1 (L1) cache. The cell area is a 62% “shrink” from the equivalent cell in IBM’s 65-nm SOI technology [29]. In scaling from 65 nm, the 12S performance cell layout was changed from butted N–P (source–drain) junctions with donutlike active shapes to nonbutted junctions and parallel active shapes. This aided scaling by improving lithographic printability and SRAM transistor variability. The 12S cell offers metal-1 bitlines for reduced capacitance. Two extra metal-2 tracks are available in the cell to facilitate array wiring. The cell’s transistor characteristics are tuned to maximize the benefit of a dual power supply [28].

The P404 ($0.404 \mu\text{m}^2$) bit cell is used for denser memories with dual power supplies such as the POWER7 Level 2

(L2) cache and Level 3 (L3) directory. Similar with the S462, the cell size is a $0.62\times$ shrink from the previous technology. The cell contains the same variability and capacitance enhancements as the S462 cell. A second version of the $0.404\text{-}\mu\text{m}^2$ cell, i.e., F404, is retuned as a performance cell for single power supply applications such as those used for ASICs libraries. Thus, the P404 and F404 share the same cell footprint, but their transistor dimensions and V_T implants differ to optimize the power, performance, and operating voltages.

The P315 cell ($0.315 \mu\text{m}^2$) is the dense cell offered in the 45-nm technology. It is also the core cell of the dual-port, two-port, and ternary-content-addressable memory cell designs. IBM ASICs are designed with a compilable P315 SRAM macro with 128 cells/bitline and 72% macro efficiency. Write-assist circuitry enables a $V_{\min}$ down to 0.8 V for 1-GHz operation [30].

In contrast, the eDRAM cell size (at $0.068 \mu\text{m}^2$) is approximately one-fifth smaller in cell area, compared with SRAM cells and allows creation of much denser memory blocks that are ideally suited for the higher levels of cache in the microprocessor. eDRAM also offers power advantages over high-performance SRAM due to the lower leakage per bit in standby and lower active power due to the lower capacitive load of the shorter wire lengths in the smaller eDRAM-based array. eDRAM standby power is reduced by 80%, and the active power is reduced by approximately 20%, compared with an SRAM solution. While small SRAM-based arrays can provide performance that is significantly faster than achievable with an eDRAM, in larger memory blocks, where performance is limited more by wiring delays than by speed of the bitcell itself, the eDRAM solution can provide system performance that is comparable to SRAM but with higher bit density and lower power

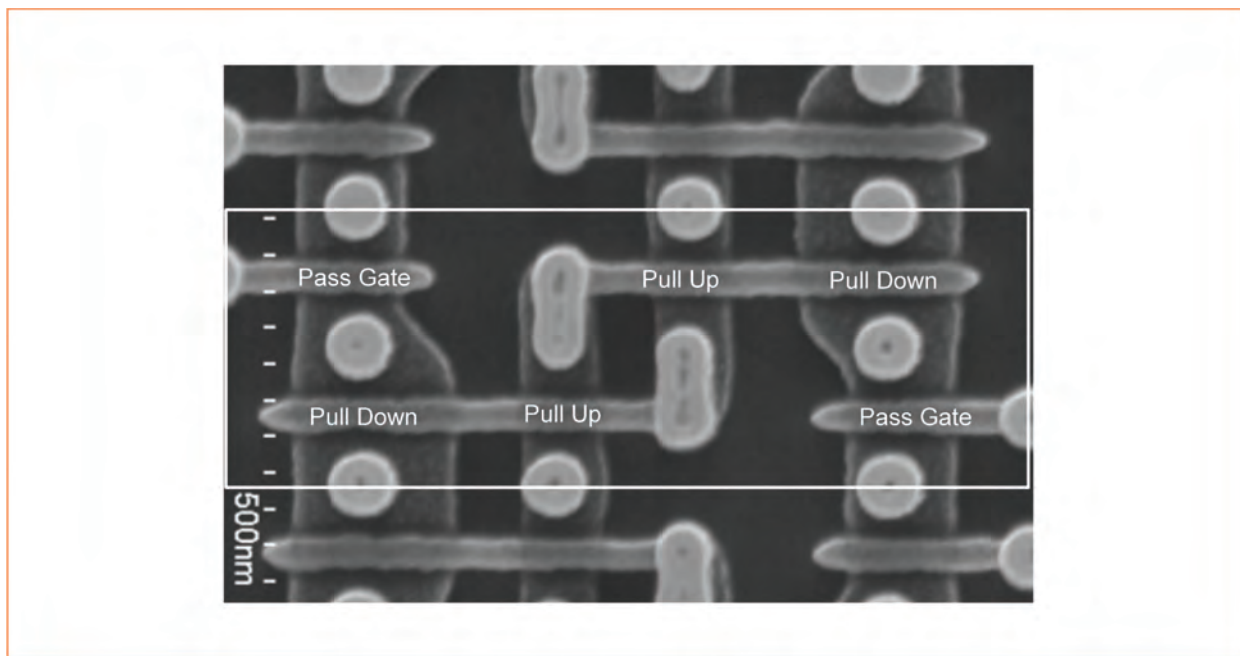


Figure 6

Scanning electron micrograph of the P404 single-port 45-nm SRAM cell used in the POWER7 L2 cache. The box outlines a unit cell. This image includes silicon, polysilicon, and contacts structures.

consumption. For the POWER7 chip, all the eDRAM L3 cache memory is implemented with a single 1-Mb macrodesign that at 1.05 V achieves 1.7-ns random cycle time and 1.35-ns data access. The IBM ASIC library provides a growable family of eDRAM macros ranging from 128 Kb to 36 Mb to allow optimizing performance and density based on application requirements.

Performance benchmarking

Incremental system performance is dependent on a number of items delivered by technology. One item is integration—not only in the traditional sense of density but also in this case embedded memory. Compared with the alternative of larger on-chip L3 cache, the embedded eDRAM solution delivers two additional cores on-chip with fewer parasitics and, thus, a system-level performance commensurate with the additional on-chip processing power. A second item, also delivered via the embedded memory technology elements, is the higher density decoupling. This decoupling provides power supply fluctuation dampening, which increases chip speeds in the range of 4%–7%.

The third item that increases system performance is more traditional transistor performance improvements. Benchmarks indicate that IBM’s 45-nm SOI technology provides a 28% performance boost from IBM’s 65-nm SOI technology, following the performance increase trend

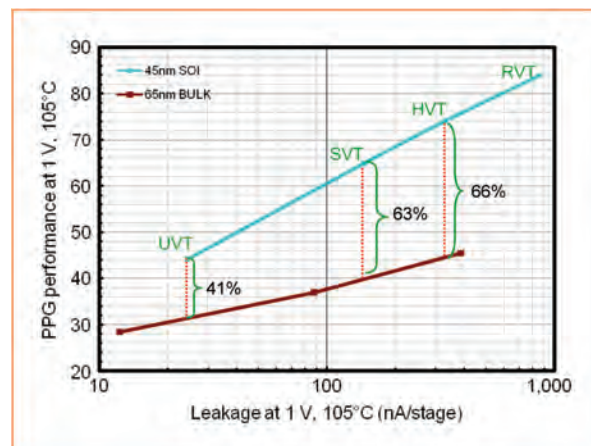


Figure 7

Cross-technology performance benchmarking: 45-nm SOI versus 65-nm bulk.

expected from technology scaling. IBM’s 45-nm SOI technology also provides superior power/performance benefit over the 45- and 65-nm bulk technology. As shown in **Figure 7**, at constant leakage, the 45-nm SOI technology shows a 66% performance advantage over the 65-nm

bulk, respectively, at a 45-nm SOI HVT leakage point (leakage per stage).

The 45-nm SOI performance benefit was benchmarked using a product performance gauge (PPG) metric. This PPG metric captures average ring delay from a suite of rings, including fan-out 3 (FO3) inverter, FO3 two-input NAND, FO3 three-input NAND, and FO3 two-input NOR rings; for example, $PPG = 0.4 \cdot \text{Delay}_{\text{INVFO3}} + 0.1 \cdot (\text{Delay}_{\text{nand2T}} + \text{Delay}_{\text{NAND2B}} + \text{Delay}_{\text{NAND3T}} + \text{Delay}_{\text{NAND3B}} + \text{Delay}_{\text{NOR2T}} + \text{Delay}_{\text{NOR2B}})$, where T means that the drain of the switching FET is tied to a ring stage output, and B means that the source of the switching FET is tied to a power supply.

Developed over time to support technology development performance goals as well as modeling benchmark accuracy, this set of structures is defined both in silicon and as an extracted netlist and has enhanced the standard technology benchmarking ring realized in silicon—a simple M1 testable FO3 inverter ring. Since the PPG circuit composition is designed to reflect a typical critical path, the accuracy of the PPG composite is targeted in the model development activity. This 45SOI compact model provides a high quality of stacked device modeling with approximately 1% discrepancy for model/hardware composite PPG performance matching. The modeling of linear current was emphasized in stack modeling [31, 32].

Conclusion

We have described an extremely versatile SOI-based technology that addresses a wide variety of applications ranging from high-end server and mainframe applications to more conventional ASIC and digital foundry applications. A distinguishing feature of this technology is the attention given to embedded memory technology that spans a wide range in the density–performance–power space, particularly the introduction of the eDRAM in SOI. This density enhancement allows the chip and system designer to include significantly more logic functions such as additional cores, integrated memory controllers, and an unprecedented bandwidth to the outside world, as described elsewhere in this issue, without increasing the chip area and while staying well ahead of the industry trend in power/performance through the use of innovative features such as immersion lithography, trench-based decoupling, and analog features. For most applications, IBM's 45-nm eDRAM-based SOI technology provides the raw circuit density or effective scaling advantage of more than one generation while providing chip-level power/performance of the next-generation technology. It achieves all this using conventional materials in a very cost-effective way for a wide variety of applications.

Acknowledgments

Complexities of technology development demand leadership and hard work from many talented individuals. The authors

would like to thank Murray Seaman, Paul Hyde, Roger Quon, Theo Standaert, Yue Tan, Dean Sciacca, Scott Springer, Mike Schmidt, Russ Robison for the leadership; Jeff Johnson for his deep insight into the technology development and coordination activities; and Fred Scholl, Ron Logan, Dave Riggs, Jang Sim, Brian Walsh, and George Gifford for their unbelievable focus in bridging this technology into the reality of POWER7 and other products.

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

References

1. R. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted MOSFETs with very small physical dimensions," *IEEE J. Solid-State Circuits*, vol. SSC-9, no. 5, pp. 256–268, Oct. 1974.
2. J. Yuan, V. Chan, M. Eller, N. Rovedo, H. K. Lee, Y. Gao, V. Sardesai, N. Kanike, V. Vidya, O. Kwon, O. S. Kwon, J. Yan, S. Fang, W. Wille, H. Wang, Y. T. Chow, R. Booth, T. Kebede, W. Clark, H. Mo, C. Ryou, J. Liang, J. H. Yang, C. W. Lai, S. S. Naragad, O. Gluschenkov, M. R. Visokay, C. Radens, S. Deshpande, H. Shang, Y. Li, N. Cave, J. Sudijono, J. Ku, and R. Divakaruni, "A 45 nm low power bulk technology featuring CarbonCo-implantation and laser anneal on 45-rotated substrate," in *Proc. Int. Conf. Solid-State Integr.-Circuit Technol.*, Oct. 2008, pp. 1130–1133.
3. M. LaPedus, "Intel drops 157-nm tools from lithography roadmap," in *E.E. Times*, May 22, 2003.
4. M. LaPedus, "Intel pushes 193-nm litho down to 15-nm," in *E.E. Times*, Jun. 19, 2009.
5. ASMLAn Introduction to Photolithography Imaging, Jul. 2007. [Online]. Available: <http://www.asml.com/asml/show.do?ctx=13559&rid=34809>
6. Y. Borodovsky, "Marching to the beat of Moore's law," *Proc. SPIE*, vol. 6153, p. 615 301, 2006. DOI: 10.1117/12.655176.
7. G. M. Wallraff, C. E. Larson, G. Breyta, L. Sundberg, D. Miller, D. Gil, K. Petrillo, and W. Pierson, "The effect of photoresist/topcoat properties on defect formation in immersion lithography," *Proc. SPIE*, vol. 6153, p. 615 31M, 2006. DOI:10.1117/12.657179.
8. M. Khojasteh, I. Popova, P. Varanasi, L. Sundberg, C. Robinson, D. Corliss, M. Lawson, G. Dabbagh, M. Slezak, M. Colburn, and K. Petrillo, "Building an immersion topcoat from the ground up: Materials perspective," *Proc. SPIE*, vol. 6519, p. 651 907, 2007. DOI: 10.1117/12.712095.
9. W. Hinsberg, G. M. Wallraff, C. E. Larson, B. W. Davis, V. Deline, S. Raoux, D. Miller, F. A. Houle, J. Hoffnagle, M. I. Sanchez, C. Rettner, L. K. Sundberg, D. R. Medeiros, R. R. Dammel, and W. E. Conley, "Liquid immersion lithography: Evaluation of resist issues," *Proc. SPIE*, vol. 5376, pp. 21–33, 2004.
10. D. P. Sanders, "Advances in patterning materials for 193 nm immersion lithography," *Chem. Rev.*, vol. 110, no. 1, pp. 321–360, Jan. 2010.
11. D. Gil, T. Bailey, D. Corliss, M. J. Brodsky, P. Lawson, M. Rutten, Z. Chen, N. Lustig, T. Nigussie, K. Petrillo, and C. Robinson, "First microprocessors with immersion lithography," *Proc. SPIE*, vol. 5754, pp. 119–128, 2005.
12. C. Robinson, J. Bright, D. Corliss, M. Guse, B. Lang, and G. Mack, "Monitoring defects at wafer's edge for improved immersion lithography performance," *Proc. SPIE*, vol. 6924, p. 692 440, 2008. DOI: 10.1117/12.776363.

13. A. Gabor, B. Liegl, M. Pike, E. Hwang, and T. Wiltshire, "The GridMapper challenge: How to integrate into manufacturing for reduced overlay error," *Proc. SPIE*, vol. 7640, pp. 764 015-1 764 015-6, 2010.
14. S. Burns, D. Pfeiffer, A. Mahorowala, K. Petrillo, A. Clancy, K. Babich, D. Medeiros, S. Allen, S. Holmes, M. Crouse, C. Brodsky, V. Pham, Y. Lin, K. Patel, N. Lustig, A. Gabor, C. Sheraw, P. Brock, and C. Larson, "Silicon containing polymer in applications for 193 nm high NA lithography processes," *Proc. SPIE*, vol. 6153, p. 615 30K, 2006. DOI: 10.1117/12.657197.
15. K. Lai, A. E. Rosenbluth, G. Han, J. Tirapu-Azpiroz, J. Meiring, A. Goehnermeier, B. Kneer, M. Totzeck, L. de Winter, W. de Boeij, and M. van de Kerkhof, "Modeling polarization for hyper-NA lithography tools and masks," *Proc. SPIE*, vol. 6520, p. 652 00D, 2007. DOI:10.1117/12.712272.
16. P. Clarke, "IBM, Toppan sign \$ 200 million photomask agreement," in *E.E. Times*, May 19, 2005.
17. J. Barth, W. Reohr, P. Parries, G. Fredeman, J. Golz, S. Schuster, R. Matick, H. Hunter, C. Tanner, J. Harig, H. Kim, B. Khan, J. Griesemer, R. Havreluk, K. Yanagisawa, T. Kirihata, and S. Iyer, "A 500 MHz random cycle 1.5 ns-latency, SOI embedded DRAM macro featuring a 3T micro sense amplifier," in *Proc. IEEE Int. Solid-State Circuits Conf., Dig. Tech Papers*, Feb. 2007, pp. 486 617.
18. G. Wang, K. Cheng, H. Ho, J. Faltermeier, W. Kong, H. Kim, J. Cai, C. Tanner, K. McStay, K. Balasubramanyam, C. Pei, L. Ninomiya, X. Li, K. Winstel, D. Dobuzinsky, M. Naeem, R. Zhang, R. Deschner, M. J. Brodsky, S. Allen, J. Yates, Y. Feng, P. Marchetti, C. Norris, D. Casarotto, J. Benedict, A. Kniffin, D. Parise, B. Khan, J. Barth, P. Parries, T. Kirihata, J. Norum, and S. S. Iyer, "A 0.127 μm^2 high performance 65 nm SOI based embedded DRAM for on-processor applications," in *IEDM Tech. Dig.*, Dec. 2006, pp. 1 4.
19. M. Horstmann, A. Wei, T. Kammler, J. Höntschel, H. Bierstedt, T. Feudel, K. Froberg, M. Gerhardt, A. Hellmich, K. Hempel, J. Hohage, P. Javorka, J. Klais, G. Koerner, M. Lenski, A. Neu, R. Otterbach, P. Press, C. Reichel, M. Trentsch, B. Trui, H. Salz, M. Schaller, H.-J. Engelmann, O. Herzog, H. Ruelke, P. Hübner, R. Stephan, D. Greenlaw, M. Raab, N. Kepler, H. Chen, D. Chidambarrao, D. Fried, J. Holt, W. Lee, H. Nii, S. Panda, T. Sato, A. Waite, S. Luning, K. Rim, D. Schepis, M. Khare, S. F. Huang, J. Pellerin, and L. T. Su, "Integration and optimization of embedded-SiGe, compressive and tensile stressed liner films, and stress memorization in advanced SOI CMOS technologies," in *IEDM Tech. Dig.*, Dec. 2005, pp. 233 236.
20. S. Narasimha, K. Onishi, H. M. Nayfeh, A. Waite, M. Weybright, J. Johnson, C. Fonseca, D. Corliss, C. Robinson, M. Crouse, D. Yang, C.-H. J. Wu, A. Gabor, T. Adam, I. Ahsan, M. Belyansky, L. Black, S. Butt, J. Cheng, A. Chou, G. Costrini, C. Dimitrakopoulos, A. Domenicucci, P. Fisher, A. Frye, S. Gates, S. Greco, S. Grunow, M. Hargrove, J. Holt, S.-J. Jeng, M. Kelling, B. Kim, W. Landers, G. Larosa, D. Lea, M. H. Lee, X. Liu, N. Lustig, A. McKnight, L. Nicholson, D. Nielsen, K. Nummy, V. Ontalus, C. Ouyang, X. Ouyang, C. Prindle, R. Pal, W. Rausch, D. Restaino, C. Sheraw, J. Sim, A. Simon, T. Standaert, C. Y. Sung, K. Tabakman, C. Tian, R. Van Den Nieuwenhuizen, H. Van Meer, A. Vayshenker, D. Wehella-Gamage, J. Werking, R. C. Wong, J. Yu, S. Wu, R. Augur, D. Brown, X. Chen, D. Edelstein, A. Grill, M. Khare, Y. Li, S. Luning, J. Norum, S. Sankaran, D. Schepis, R. Wachnik, R. Wise, C. Warm, T. Ivers, and P. Agnello, "High performance 45-nm SOI technology with enhanced strain, porous low- k BEOL, and immersion lithography," in *IEDM Tech. Dig.*, Dec. 2006, pp. 1 4.
21. H. M. Nayfeh, S. Jeng, S. Narasimha, S. Butt, R. Pal, A. Waite, K. Tabakman, J. B. Johnson, J. Liu, J. Holt, T. Adam, A. Madan, and A. Domenicucci, "Hole transport in nanoscale p-type MOSFET SOI devices with high strain," in *Proc. 65th DRC*, 2007, pp. 51 52.
22. C.-H. Chen, T. L. Lee, T. H. Hou, C. L. Chen, C. C. Chen, J. W. Hsu, K. L. Cheng, Y. H. Chiu, H. J. Tao, Y. Jin, C. H. Diaz, S. C. Chen, and M.-S. Liang, "Stress memorization technique (SMT) by selectively strained-nitride capping for sub-65 nm high-performance strained-Si device application," in *VLSI Symp. Tech. Dig.*, 2004, pp. 56 57.
23. X. Chen, S. Fang, W. Gao, T. Dyer, Y. W. Teh, S. S. Tan, Y. Ko, C. Baiocco, A. Ajmera, J. Park, J. Kim, R. Stierstorfer, D. Chidambarrao, Z. Luo, N. Nivo, P. Nguyen, J. Yuan, S. Panda, O. Kwon, N. Edleman, T. Tjoa, J. Widodo, M. Belyansky, M. Sherony, R. Amos, H. Ng, M. Hierlemann, D. Coolbough, A. Steegen, I. Yang, J. Sudijono, T. Schiml, J. H. Ku, and C. Davis, "Stress proximity technique for performance improvement with dual stress liner at 45 nm technology and beyond," in *VLSI Symp. Tech. Dig.*, 2006, pp. 60 61.
24. A. Asenov, A. Brown, J. Davies, S. Kaya, and G. Slavcheva, "Simulation of intrinsic parameter fluctuations in decananometer and nonometer-scale MOSFETs," *IEEE Trans. Electron Devices*, vol. 50, no. 9, pp. 1837 1852, Sep. 2003.
25. S. Burns, M. Burkhardt, D. Goldfarb, N. Lustig, D. Pfeiffer, M. J. Brodsky, A. Clancy, and D. Medeiros, "Trilayer materials for 193 nm immersion lithography," *J. Photopolym. Sci. Technol.*, vol. 20, no. 5, pp. 679 686, Sep. 2007.
26. A. Grill and V. Patel, "Ultralow- k dielectrics prepared by plasma-enhanced chemical vapor deposition," *Appl. Phys. Lett.*, vol. 79, no. 6, pp. 803 805, Aug. 2001.
27. S. Sankaran, S. Arai, R. Augur, M. Beck, G. Biery, T. Bolom, G. Bonilla, O. Bravo, C. Chanda, M. Chae, F. Chen, L. Clevenger, S. Cohen, A. Cowley, P. Davis, J. Demarest, J. Doyle, C. Dimitrakopoulos, L. Economikos, D. Edelstein, R. Filippi, J. Fitzsimmons, N. Fuller, S. Gates, S. Greco, A. Grill, S. Grunow, R. Hannon, K. Ida, D. Jung, M. Kelling, T. Ko, K. Kumar, C. Labelle, H. Landis, M. Lane, W. Landers, M. Lee, W. Li, E. Liniger, X. Liu, J. Lloyd, W. Liu, N. Lustig, K. Malone, S. Marokkey, G. Matusiewicz, P. S. McLaughlin, P. V. McLaughlin, S. Mehta, I. Melville, K. Miyata, B. Moon, S. Nitta, D. Nguyen, L. Nicholson, D. Nielsen, P. Ong, K. Patel, V. Patel, W. Park, J. Pellerin, S. Ponoath, K. Petrarca, D. Rath, D. Restaino, S. Rhee, E. Ryan, H. Shoba, A. Simon, E. Simonyi, T. Shaw, T. Spooner, T. Standaert, J. Sucharitaves, C. Tian, H. Wendt, J. Werking, J. Widodo, L. Wiggins, and T. Ivers, "A 45 nm CMOS node Cu/low- k /ultra low- k PECVD SiCOH($k = 2.4$) BEOL technology," in *IEDM Tech. Dig.*, 2006, pp. 1 4.
28. K. J. Kim, J. M. Youn, S. B. Kim, J. H. Kim, S. H. Hwang, K. T. Kim, and Y. S. Shin, "A novel 6.4 μm^2 full-CMOS SRAM cell with aspect ratio of 0.63 in a high-performance 0.25 μm -generation CMOS technology," in *VLSI Symp. Tech. Dig.*, Jun. 9 11, 1998, pp. 68 69.
29. D. W. Plass and Y. H. Chan, "IBM POWER6 SRAM arrays," *IBM J. Res. & Dev.*, vol. 51, no. 6, pp. 747 756, Nov. 2007.
30. D. P. Wang, H. J. Liao, H. Yamauchi, Y. H. Chen, Y. L. Lin, S. H. Lin, D. C. Liu, H. C. Chang, and W. Hwang, "A 45 nm dual-port SRAM with write and read capability enhancement at low voltage," in *Proc. IEEE Int. SOC Conf.*, Sep. 26 29, 2007, pp. 211 214.
31. K. V. Arim, C. Pacha, K. Hofmann, T. Schulz, K. Schröder, and J. Berthold, "An effective switching current methodology to predict the performance of complex digital circuits," in *IEDM Tech. Dig.*, Dec. 2007, pp. 483 486.
32. X. Yu, S. Han, N. Zamdmer, J. Deng, E. J. Nowak, and K. Rim, "Improved effective switching current (IEFF+) and capacitance methodology for CMOS circuit performance prediction and method-to-hardware correlation," in *IEDM Tech. Dig.*, Dec. 2008, pp. 1 4.

Received March 3, 2010; accepted for publication November 16, 2010

Subramanian (Subu) Iyer *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (ssiyer@us.ibm.com).* Dr. Iyer obtained his B.Tech. degree in electrical engineering at the Indian Institute of Technology, Bombay, and his M.S. and Ph.D. degrees in electrical engineering at the University of California at Los Angeles. He is Distinguished Engineer and Chief Technologist for the Semiconductor Research and Development Center, IBM Systems and Technology Group, and is responsible for technology development strategy, embedded memory, and 3-D integration. Until recently he was Director of 45-nm CMOS development. He joined the IBM Thomas J. Watson Research Center in 1981 and was manager of the Exploratory Structures and Devices Group until 1994, when he founded SiBond LLC to develop and manufacture silicon-on-insulator materials. He has been with the IBM Microelectronics Division since 1997. He holds more than 40 patents and has received 21 Invention Plateau awards at IBM and is a Master Inventor. He received the Distinguished Alumnus Award from the Indian Institute of Technology, Bombay, in 2004. Dr. Iyer has authored more than 150 articles in technical journals and several book chapters, and he has co-edited a book on bonded SOI. He has served as an Adjunct Professor of Electrical Engineering at Columbia University. Dr. Iyer is a Fellow of IEEE and a Distinguished Lecturer of the IEEE and Chair of the mid-Hudson chapter of the Electron Device Society.

Gregory Freeman *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (freemang@us.ibm.com).* Dr. Freeman received his Ph.D. degree from Stanford University in 1991 and has been with IBM in East Fishkill, NY, since 1991 where he has been involved in characterization, HBT design, and SOI CMOS technology development. Dr. Freeman has authored or coauthored more than 60 publications and has approximately 26 patents. Currently he is manager of the 45-nm and 22-nm SOI device design department at IBM.

Colin Brodsky *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (cbrodsky@us.ibm.com).* Dr. Brodsky joined the IBM Semiconductor Research and Development Center in 2001 after earning his Ph.D. degree in chemical engineering from the University of Texas at Austin and his B.S. degree in chemical engineering from Carnegie Mellon University. Since joining IBM, he has been responsible for the development of a broad range of lithography processes as a process development engineer for the 90-nm and 65-nm technology nodes, and since 2006, as manager of the SOI lithography development team for the 45-nm and 32-nm technology nodes. His technical interests have included collaborative development of new experimental designs and characterization techniques that exploit advanced process yield learning methodologies, productivity improvements with the first immersion lithography manufacturing toolset, and implementation of systematic patterning variability analysis techniques to improve process window optimization and centering efforts.

Anthony I. Chou *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (tonychou@us.ibm.com).* Dr. Chou received his Ph.D. degree in electrical engineering from the University of Texas at Austin in 1997. He joined Texas Instruments in Dallas, Texas, in 1997 and worked on Flash memory devices and technology. He joined IBM in East Fishkill, NY, in 1998, and developed several generations of gate oxide for advanced CMOS technologies. Since 2006, he has worked on SOI device design, focusing on analog FET and passive device development. Dr. Chou is author or coauthor of several patents and technical papers.

Dan Corliss *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (corlissd@us.ibm.com).* Mr. Corliss has been working on and leading lithography development programs for the past 25 years. He joined IBM in 1998 and is currently an STSM and Program Manager for imaging systems development. In 2004 he

became IBM's Immersion Lithography Program manager and was chartered with taking immersion lithography from concept to manufacturing reality. He continues to mentor the team that now provides immersion lithography solutions for IBM and its alliance partners.

Sameer H. Jain *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (sjain@us.ibm.com).* Dr. Jain received his Ph.D. degree in applied physics from Stanford University, Palo Alto, CA, in 2005. He subsequently joined the CMOS Performance Solutions team at the IBM Semiconductor Research and Development Center where he worked on the use of millisecond laser annealing to improve performance in IBM's CMOS technologies. In 2007 he joined the 45-nm SOI team focusing on extending the range of the high-performance technology to low-powered FETs used primarily for ASICs and foundry applications. He is currently leading the device development for IBM's 20-nm low-power bulk technology.

Naftali Lustig *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (lustig@us.ibm.com).* Dr. Lustig received a B.S. degree in physics from the University of California at Berkeley and a Ph.D. degree in physics from Pennsylvania State University. He joined IBM in 1986 at the Thomas J. Watson Research Center as a Postdoctoral Fellow, characterizing thin-film transistors for liquid crystal displays. In 1988 he joined IBM East Fishkill working on ohmic contacts to GaAs MESFETs. His involvement with on-chip interconnects started in the early 1990s in the area of oxide planarization and chemical-mechanical planarization (CMP) endpoint detection. He carried out pioneering work in copper CMP, copper process integration, and design rules for damascene wiring. He has contributed to various aspects of the back-end-of-line (BEOL) in every technology node since then. Currently he is a Senior Engineer working on the 22-nm node. Dr. Lustig holds 16 U.S. patents, authored more than 50 publications, and is a member of the American Physical Society.

Vincent McGahay *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (mcgahay@us.ibm.com).*

Shreesh Narasimha *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (narasimha@us.ibm.com).*

James Norum *IBM Systems and Technology Group, 2070 Route 52, Hopewell Junction, NY 12533 USA (norum@us.ibm.com).* Mr. Norum received his B.S. degree in electrical engineering from Yale University in 1987 and his M.Eng. degree in electrical engineering from Cornell University in 1990. He is a member of the IEEE. He joined IBM in 1987 and worked on advanced bipolar process integration, before joining reactive ion etch (RIE) manufacturing engineering. He has managed plasma equipment engineering, RIE manufacturing engineering, embedded DRAM process integration, including the first introduction of trench-based eDRAM into the market place, and bulk front-end-of-line (FEOL) process integration. Since 2006, Mr. Norum has managed SOI FEOL process integration, including the 45-nm and 22-nm technology nodes.

Karen A. Nummy *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (knummy@us.ibm.com).* Ms. Nummy received her S.B. and S.M. degrees in materials science from Massachusetts Institute of Technology, Cambridge, in 1981. She joined the development organization in IBM East Fishkill in 1981 and over the course of her career has worked in process integration development in many areas, including Si bipolar, SiGe bipolar, and CMOS devices. Her most recent assignment was the lead eDRAM integrator for the 45-nm 12S technology, and she is currently the lead eDRAM integrator for 22S.

Paul Parries *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (parries@us.ibm.com).* Dr. Parries received his B.S. and Ph.D. degrees in electrical engineering from the University of Minnesota prior to joining IBM in Burlington, VT, in 1985. He is a Senior Technical Staff Member working in embedded DRAM technology and product development in East Fishkill, NY. He has more than 20 years of experience working on commodity DRAM and embedded DRAM technologies and has expertise in process development, DRAM process integration, and circuit design and characterization.

Sujatha Sankaran *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (sujatha@us.ibm.com).*

Chris D. Sheraw *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (csheraw@us.ibm.com).* Dr. Sheraw received his Ph.D. degree in electrical engineering from The Pennsylvania State University in 2003. He then joined the FEOL Process Integration team at the IBM Semiconductor Research and Development Center where he has worked on SOI CMOS technology development at the 90-nm and 65-nm nodes, and as the lead FEOL SOI Process Integration Engineer for the 45-nm node. He is now working on 22-nm CMOS technology development. His interests include semiconductor process development, thin film electronics, and nanotechnology.

Pushkara Rao Varanasi *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (varanasi@us.ibm.com).* Dr. Varanasi is a Senior Technical Staff Member and Research and Development Manager in the IBM Semiconductor Research and Development Center. He is responsible for patterning materials strategy and execution, as well as for all joint development partnerships with photoresist suppliers. His team has pioneered numerous patterning materials innovations, in collaboration with the IBM Almaden materials research team. Dr. Varanasi joined the Materials Group in 1996 after working in the chemical industry for several years. His education and academic training was in organic photochemistry at Columbia University.

Geng Wang *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (wanggeng@us.ibm.com).* Dr. Wang received his B.S. degree in physics from Peking University, China, in 1995, and M.S. and Ph.D. degrees in electrical engineering from the University of Texas at Austin, in 1997 and 2001, respectively. He has been with the IBM Microelectronics division since 2001. As a TCAD engineer, he worked on the 110-nm and 90-nm vertical access transistor DRAM cell and studied the scalability of planar access devices for DRAM cells. In 2003, he worked on the access device design of IBM's first generation of high-performance eDRAM, which eventually led to the successful integration of eDRAM on the POWER7 multicore processor. He is a Senior Engineer and continues to focus his interest on scaling high-performance SOI-based embedded DRAMs.

Mary E. Weybright *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (weybrigh@us.ibm.com).* Dr. Weybright received her B.S. degree in electrical engineering from Michigan State University and her M.S. and Ph.D. degrees in electrical engineering from Stanford University. She joined the IBM Semiconductor Research and Development Center in 1996 and has since worked on device and front-end-of-line process integration development for DRAM, SRAM, and embedded FLASH technologies. She led the early SRAM cell development for 45-nm SOI technology. Dr. Weybright is a Senior Engineer and has reached the fifth IBM Invention Achievement plateau.

Xiaojun Yu *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (xyu@us.ibm.com).* Dr. Yu received his B.S. and M.S. degrees in material science and engineering from Tsinghua University, Beijing, in 1998 and 2000, respectively. He then

obtained his M.S. degree in electrical engineering and Ph.D. degree in material science and engineering from Stanford University in 2005. From 2005 to 2006, he worked on ultrahigh-density multilevel NAND Flash circuit design at Micron Technology, Inc. In 2006, he joined IBM Semiconductor Research and Development Center as Advisory Engineer, where he has been engaged in product leakage prediction methodology and product power/performance optimization, as well as a device variability study for various core technologies.

Emmanuel Crabbe *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (crabbe@us.ibm.com).*

Paul Agnello *IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (agnello@us.ibm.com).* Dr. Agnello is a Distinguished Engineer and 32-nm SOI Chief Technologist at the IBM Semiconductor Research and Development Center at Hopewell Junction, NY. He received his B.S., M.S., and Ph.D. degrees, all from the Electrical, Computer and Systems Engineering Department of Rensselaer Polytechnic Institute, and joined the IBM Thomas J. Watson Research Center in 1988 and the IBM Microelectronics Division in 1992. Over his career at IBM, he has worked on developing and bringing to manufacturing more than ten generations of high-performance microprocessor technology. Currently he manages the High Performance CMOS Integration and Device project that is responsible for the 32-nm node high-performance microprocessor technology. He has authored or coauthored more than 60 publications and holds more than 15 U.S. patents.

IBM POWER7 processor circuit design

D. F. Wendel
J. Barth
D. M. Dreps
S. Islam
J. Pille
J. A. Tierno

The IBM POWER7® processor contains many innovative circuit ideas that enable advanced architectural features. A high-density embedded dynamic random access memory is used to provide 32 MB of level-3 cache. Improved input/output (I/O) links provide up to 50 GB/s of I/O bandwidth. An innovative phase-locked loop design allows for dynamic, per-core frequency variation, and unique multiport techniques for register files as well as six-transistor cell-based memory cells to support the superscalar multithreaded out-of-order processor.

Introduction

Like all modern high-end processors, the IBM POWER7* processor is a highly complex multicore multithreaded machine that pushes the technology of today to its limits [1, 2]. Some of the limitations to processor performance come from the ability of the processor package to provide power to the processor, dissipate heat, and provide enough input/output (I/O) bandwidth to maintain the processor cores fed with data; this is what is known as the “three sides” problem. Because chips have only two sides, i.e., top and bottom, the chip architects have to trade off power, heat, and I/O to fit in those two available sides.

Part of the chip architects’ job is to also reduce the need for power, heat dissipation, and I/O bandwidth at a given performance level. Large caches can be used to increase cache hit ratios; core clock frequency can be decreased on idle or less heavily loaded cores used to reduce power consumption and heat generation; and high-speed I/O can make better use of the available package pins. These are some examples of building blocks circuit designers can provide to create processors that can be balanced at a higher level of performance.

On-chip caches occupy a significant percentage of the area of most processor dies. A larger cache typically results in better performance, but in the multicore world, a larger cache may imply a reduction in the number of cores, with a corresponding performance decrease. Increasing the memory density is the only option to achieve larger caches, given that processors use the largest economically feasible die area. On-chip memory has been the target of area optimization for

so long that it is difficult to find a way to make a meaningful reduction in memory area, except by changing the memory technology from static random access memory (SRAM) to dynamic RAM (DRAM). This is the approach taken in the POWER7 processor, where 32 MB of on-chip level 3 (L3) cache is implemented with 1 Mb of embedded DRAM (eDRAM) blocks. The equivalent amount of SRAM would have an area $3\times$ larger than that of eDRAM.

Frequency variation capability is implemented on a per-core basis, using an all-digital phase-locked loop (DPLL). Among other advantages, the DPLL uses a minimum of analog devices and circuits and is therefore less sensitive to process and ambient variations. More importantly, the digital nature of the DPLL makes this circuit ideal for the introduction of digitally managed frequency profiles. Under program control, the DPLL can create frequency ramps that minimize the effect of power supply current variations on the power supply voltage.

Each core in the POWER7 processor can support four simultaneous hardware threads (SMT4). Register files must have large numbers of ports to supply data to all of the threads. Register files use double-cell architecture with double-pumped write ports that enable fast thread switching while still maintaining low power and area overhead. The level 1 (L1) cache employs a memory element based on the six-transistor (6T) memory cell but implements two read ports plus one separate write port.

System I/O (to other processors and peripherals) is the third generation of the Elastic Interface (EI3) that was previously employed in the IBM POWER5* and POWER6* processors. EI3 provides up to 50 GB/s of data bandwidth, which is twice as much as the best previously reported numbers [3]. Main memory is attached to the processor

Digital Object Identifier: 10.1147/JRD.2011.2143150

©Copyright 2011 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied by any means or distributed royalty free without further permission by computer-based and other information-service systems. Permission to republish any other portion of this paper must be obtained from the Editor.

0018-8646/11/\$5.00 © 2011 IBM

through eight direct media interface (DMI) channels, or twice as many as have been previously reported [4]. I/O bandwidth is a key component of the POWER7 performance, and circuit design has focused on implementing that bandwidth at a low area and power budget.

Circuit overview

The POWER7 chip contains more than 1.2 billion transistors on a 567-mm² die implemented in 45-nm complementary metal–oxide–semiconductor (CMOS) silicon-on-insulator (SOI) [5]. The processor has eight cores with four simultaneous hardware threads per core and a total of 32 MB of L3 cache.

Most digital circuits are implemented with static CMOS logic. Dynamic circuit structures are exclusively employed in SRAM arrays and register files. Logic transistors are available with three different threshold voltages, which enables performance versus standby power tuning. Standard cells are designed so that the transistor type can be exchanged without any changes to the circuit floor plan or wiring. Device type is optimized in postlayout timing runs.

All digital circuits, whether they are data flow, synthesized logic, register files, or memory arrays, are analyzed and verified at the transistor level. Transistor-level static timing analysis and logic equivalence are used to ensure circuit correctness.

A 0.404 μm^2 6T memory cell is used for dense SRAM arrays, such as the 256-KB per-core level-2 cache, or the L3 cache directory. The dense arrays are operated at half the core frequencies with one read/write port and 32 cells on the local bit lines (LBLs). The high-performance core arrays utilize a 0.462- μm^2 6T memory cell. The macro architecture provides up to two independent single-ended read ports and one write port. A register file compiler creates multiported register files with port configurations from one read and one write port to four read and three write ports. The resulting register file can be tested using scan or array built-in self-test. Dense dual-port arrays, with one read and one write port, are created using eight-transistor dense memory cells.

Pulse-mode latches are the default latch type; they are used to minimize power and insertion latency and provide a small transparent window to average out hardware-introduced path uncertainties over two cycles. A single global clock is distributed to each core by a tuned clock grid, guaranteeing less than 5-ps local clock skew. Local clock buffers split the clock into two phases. The latches are soft error rate (SER) hardened through the use of stacked devices in the latch path to hold the data. This topology, together with the SOI SER advantage, results in a 30 $\times$ improved robustness against soft error events, compared to a standard bulk CMOS implementation.

High-speed I/O

The POWER7 high-speed I/O design significantly changed relative to POWER6. On POWER7, all I/Os are on the

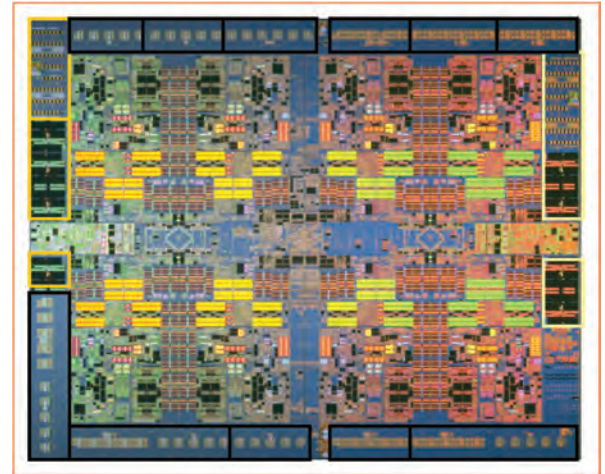


Figure 1

IBM POWER7 die showing the peripheral I/O footprint.

perimeter ring with 185.6- μm controlled collapse chip connection (C4) pitch to allow the most flexible and lowest crosstalk breakout when using organic single-chip modules or glass-ceramic multichip modules (MCMs) (see **Figure 1** for reference). The area of the ring is bounded by the C4 footprint, rather than the silicon circuit area. All available area was backfilled with deep trench (DT) capacitors (DTCAPs) connected between power supply and ground, in order to suppress the I/O voltage rail noise to 40 mV.

The system I/O links are an incremental extension of the POWER6 EI3. These links can have up to 3.2 Gb/s per lane signaling rate while also being backward compatible with POWER6 systems. There are a total of seven system links, each 8 bytes wide in either direction. These I/Os can drive short connections on MCMs all the way to 40-in-long flex cables connecting separate drawers.

The memory interface (i.e., DMI) was converted from EI3 on POWER6 to a low-power differential approach running at a rate of 6.4 Gb/s per lane. There are a total of eight ports, each having 2 bytes for read and 1 byte for write. DMI limits the use of current-mode logic circuits to clock distribution and generation and receiver peaking.

L3 eDRAM implementation

The 32-MB on-chip L3 cache is constructed using multiple instances of 1-Mb DRAM arrays, eliminating the delay, area, and power associated with an off-chip memory interface. System performance is improved, whereas power, cost, and SER vulnerability are reduced. **Figure 2(a)** shows a scanning electron micrograph of the 45-nm SOI DRAM device and DTCAP [6, 7]. DT offers 25 $\times$ more capacitance than planar structures and is also utilized to reduce on-chip power supply noise.

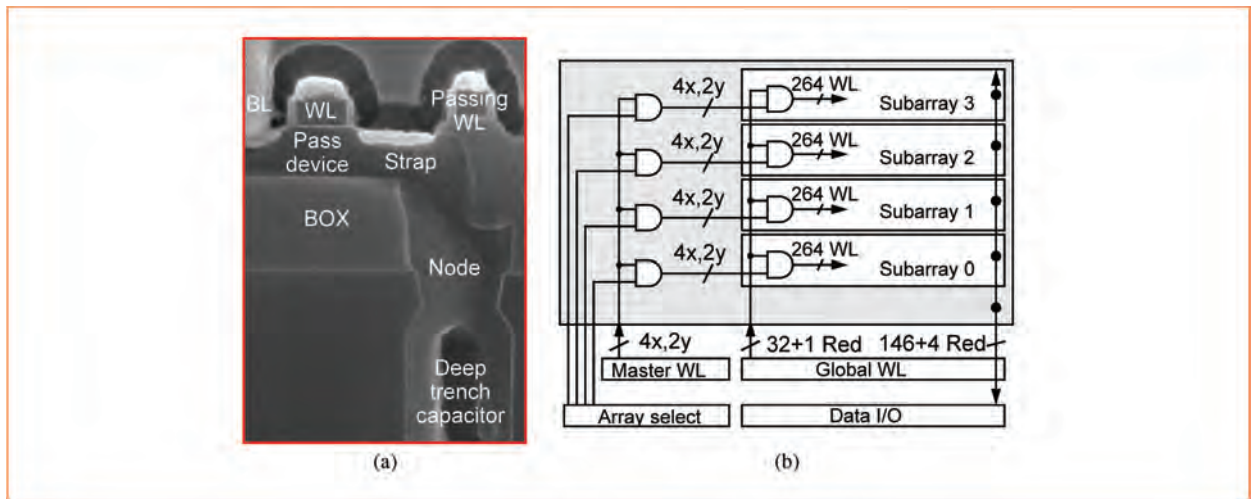


Figure 2

(a) SOI DT cell and (b) eDRAM-macro architecture.

Bulk trench technologies must deal with a parasitic leakage created by the vertical device formed between the node and trench plate conductor, gated by the trench sidewall. To mitigate this leakage, bulk technologies must form a thick oxide collar at the top of the trench. In SOI, the bulk P-well is replaced by the buried oxide (BOX), isolating the node from the trench plate and precluding the need for a collar. Eliminating the collar not only reduces process complexity but also increases the cross-sectional area of the trench, further reducing parasitic resistance to the storage capacitor. These factors combine to drop the overall eDRAM process cost adder from 15% for bulk to 7% for SOI. In addition, the absence of wells in SOI realizes lower source-drain capacitances and enables layout area reduction by eliminating well tie-downs and allowing smaller n-field-effect transistor (FET)-to-p-FET spacing.

The 1-Mb macro [see **Figure 2(b)**] is composed of four 292-Kb subarrays (264 word lines $\times$ 1,200 bit lines) above a consolidated control logic and 146 I/O regions, where input and outputs are pipelined. The row address path is duplicated to allow concurrent refresh of a second subarray. One-of-eight, one-hot, column signals offer late selection to support set-associative cache designs.

Conventional techniques for improving DRAM performance involve reducing the bit-line length; however, this increases the area overhead because of additional sense amplifiers, bit-line twisting, and reference circuits. Sense amplifier area is further degraded with body-tied SOI devices, which is required to prevent history-induced sense amplifier mismatch on small-signal long bit-line architectures. The μ SA architecture [8] was developed to provide high-performance sensing without incurring

overhead associated with conventional techniques. This hierarchical scheme relies on a high transfer ratio during a read, in order to create a large voltage swing on an LBL, large enough to be sampled with the single-ended μ SA. The single-ended nature of this sensing scheme allows a relaxed metal-1 pitch, reducing the line-to-line capacitance by a factor of 3. The short bit line, relaxed pitch, and low SOI junction capacitance combine for an LBL capacitance of only 3.5 fF. When coupled with an 18-fF cell, an 84% transfer ratio is realized, ideally producing over 800 mV of signal—more than enough to turn on a 250-mV transistor. The array operates at 500 MHz with a 1.7-ns random cycle at 1.05 V, with data access in 1.35 ns.

Digital phase-locked loop

With the number of cores in modern processors growing to eight and more, we need to address power management at the core level. One common method used to trade off power against performance is to change the clock frequency of the processor, in response to workload-dependent variations for a single core and across multiple cores [9]. In order to be practical in a real system environment, circuits that are used to enable frequency variations must have low area and power overhead. Moreover, to extract the full value of per-core dynamic frequency synthesis capability, it must be possible to change the frequency while the core is executing code. This drives a set of requirements for the variable frequency generator: 1) a near-continuous set of achievable output frequencies; 2) a controllable frequency slew rate with no skipped cycles, to limit the power supply drop caused by di/dt ; and 3) no short cycles ever generated, to prevent timing hazards. In the following paragraphs, a variable

the loop filter. In the modified PFD, unlike the previously reported BB PFD design, cycle slips are detected for downstream use. Note that when a cycle slip occurs, a second edge of one of the clocks arrives without an intervening edge of the other clock. In this case, within the modified PFD, the second edge of the fast clock arrives when the corresponding edge detector latch is still set, as reset can only be asserted after both clock edges have arrived. In this case, the corresponding faster signal will be set, indicating which clock is running faster. Because the faster signals are asserted every time a cycle slip happens, they are asserted with a frequency that is proportional to the difference in frequency between the two clocks. Once the DPLL is locked, the two clock frequencies are the same, and the faster signals are automatically deasserted.

During the frequency ramp, the clock generator never produces a short cycle. Three distinct regions are clearly identifiable: frequency acquisition, phase acquisition, and phase lock. In the frequency acquisition region, the generator output follows the frequency multiplier ramp. Every time that a cycle slip occurs between the reference and the feedback clock, the output frequency is changed by about 30 MHz. Once the frequency multiplier is stable and the generator output is close to the target frequency, cycle slips stop occurring. In this phase acquisition region, the final phase and frequency are attained using the integrator in the DPLL loop filter. Proper selection of DPLL loop filter constants ensures that no overshoot or undershoot in cycle time is present in the transient. The demonstrated combination of the controlled frequency slew rate and the absence of short cycles proves that the proposed DPLL-based variable frequency generator can be used to dynamically adjust the frequency while the core is executing code.

The DCO has a measured tuning range of 800 MHz–12 GHz over the full process, voltage, and temperature range. The DCO output is divided by two to improve the duty cycle of the clock provided to the processor core logic. Root-mean-square period jitter is 1 ps at 5 GHz and 6 ps at 1 GHz. The DPLL has an area of $200\ \mu\text{m} \times 350\ \mu\text{m}$, half of which is occupied by the voltage regulator for the DCO.

Double-pumped multiported register files

The vector-scalar unit uses four instances of a large register file, i.e., the vector register file (VRF). Like the general-purpose register (GPR) in the fixed-point unit, the VRF uses a double-pumped write technique supporting four writes per cycle via two physical write ports. Each instance of the VRF macro has 32 data bits, one implicit or undefined indication bit, and four error-correcting-code bits. With each cell having two storage elements, there are a total of 74 bits in each VRF instance. Each instance holds 172 functional entries plus two redundant entries. The register renaming logic of the instruction sequence unit ensures that the spare entries are never allocated. The VRF supports six read

and four write operations per cycle, whereas the GPR supports four read and four write operations. **Figure 4** [1] shows the double-bit 6R 4W VRF cell with the associated timing diagram. One addressable location holds two subcells. Subcells 0 and 1 store thread 0/1 data and thread 2/3 data, respectively. The six read ports are divided into two groups of three operands, with each set belonging to one thread. The thread select signal that selects subcell 0 or 1 is done inside the cell via the two-way built-in multiplexer. The multiplexing inside the cell is done via two OR-AND-INVERT(OAI22) operations. Read ports 0–2 are selected via the first OAI22, and read ports 3–5 via the second one. A duplication of the register files, to support up to four threads, is avoided with this configuration of double-bit cell and cell internal MUX.

The two physical write ports per storage element are implemented via two pairs of pass gates, connecting the subcells to share the write data lines. The selections are done via individual write word lines, which can be turned on once per cycle for a particular address. The write operations are separated into the c1 and c2 phase (first and second half of the global clock). Read happens in the c2 phase. Clear separation between c1/c2 writes and c1 writes/c2 reads is required. Architecturally, simultaneous c2 write and read to the same cell is not supported. Dual-bit cells and double-pumped write combined shorten the LBL and the global bit lines, which results in a power saving of approximately 25% of the total macro power and speed up the read operation by 5% as a result of the shorter bit lines.

Multiport SRAM implementations based on 6T SRAM cells

Increasing demand for parallelism due to out-of-order and multithreading requires fast arrays with multiport capabilities while maximizing the density. For the 32-KB L1 data cache of the POWER7 microprocessor core, a new read/write scheme has been developed that concurrently supports two independent read and one banked write operation using a standard six-device SRAM cell [11]. Organized in banks of 16 cells each, the two read operations can be fully independent in any of these banks, including two reads within the same bank, even the same cell. A bank selected for write is blocked for any read operation. If read and write collide within the same bank, a specific collision control circuitry provides write-over-read priority. Each read port provides 4 bytes from 1 of 256 locations, whereas the double-bandwidth write operation provides individual control of 8 bytes to 128 locations.

The read/write decoding scheme of the L1 data cache is shown in **Figure 5** [11]. With only 16 SRAM cells connected to the bit line, the load is low enough so that the $0.462\text{-}\mu\text{m}^2$ SRAM can be tuned in favor of performance. The two pass-gate devices are connected to two separate word lines

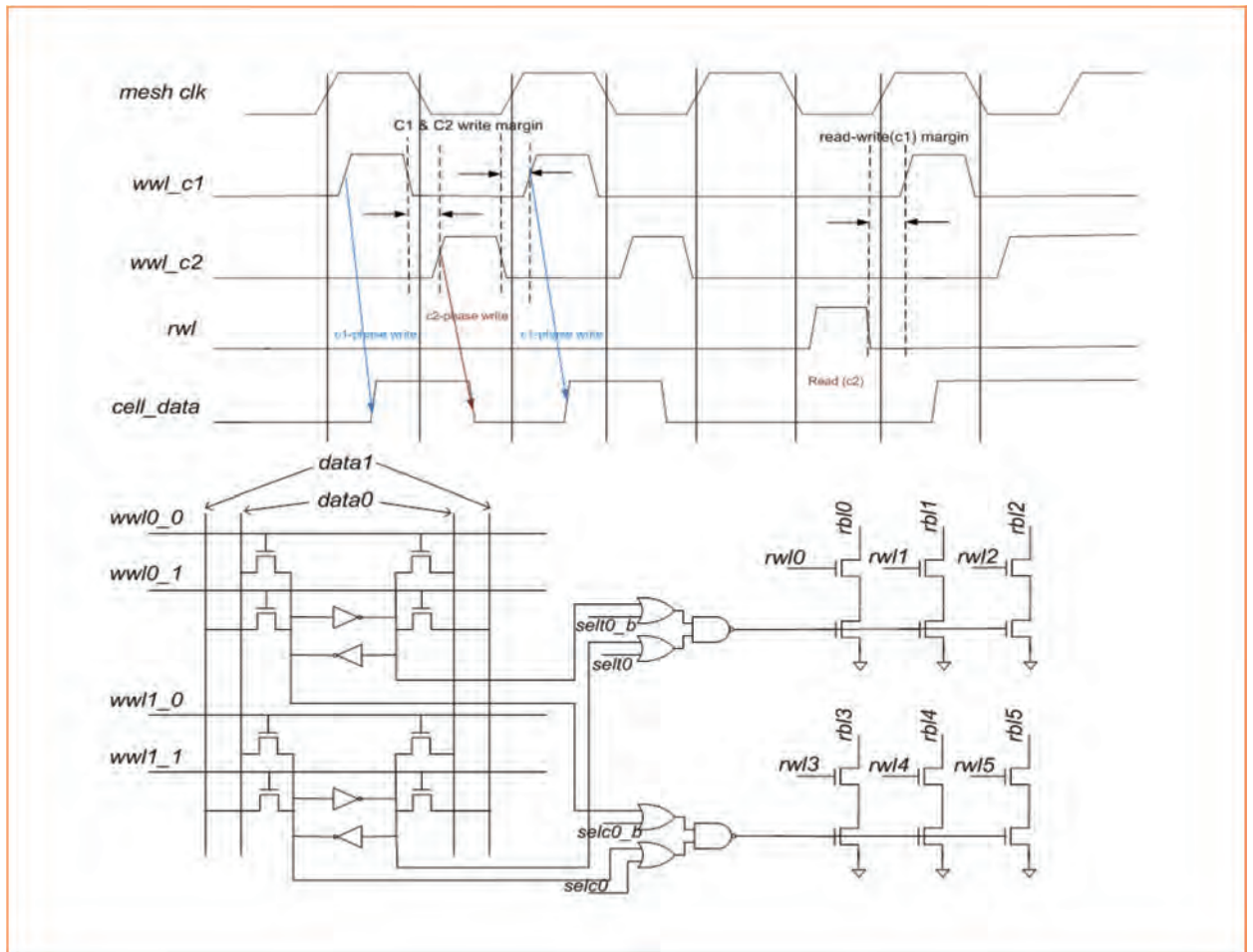


Figure 4
Double-bit and double-pumped register file cell with the associated read write timing diagram (©2011 IEEE).

(wl_t, wl_c) and LBLs (bl_t, bl_c). Single-ended reads are initiated by activating a word line connected to one of the pass gates. For a write operation, both word lines of a given cell are active for a differential write.

The two-stage decoder is organized in a bank select (msb) and a row select within a bank (lsb). In the read/write bank control stage, the collision case is handled. If a bank *m* is not selected for a write (wr_{msb}(*m*) = 0), the read address can take control (rd_{msb}(*m*) = rd_{lsb}(*n*) = 1), with each of the two read ports controlling a different word line. If both read addresses are the same, the two word lines wl_t and wl_c of the same cell are selected for read, which is supported. If a bank *m* is selected for a write (wr_{msb}(*m*) = 1), it takes control independent of the read address by overwriting rd[0, 1]_{msb}(*m*) of both read ports to zero, which blocks the read addresses for this bank. Since the write msb and lsb signals are connected to the read/write

bank control circuits of both read ports, it is guaranteed that both word lines of the same cell entry (wl_t and wl_c) are now selected, which is a prerequisite to guarantee a robust differential SRAM cell write scheme. The LBL precharge devices need to be disabled during a read or write access; therefore, the bank select signals of the two read and one write port are OR-ed together, controlling the precharge devices.

Figure 6 [11] shows the local read and write circuit to support two read and one write operation. It serves two banks 0 and 1 (to the bottom and top), each connected to 16 cells.

To read the true port in bank 1, the pass gate of a cell in that bank is selected by the word line wl_t, pulling down LBL blt1, if the content of the selected cell is zero (prch1_b turned off). Together with rdsel1_b being low, the signal propagates and raises node *dt*, which pulls down the global

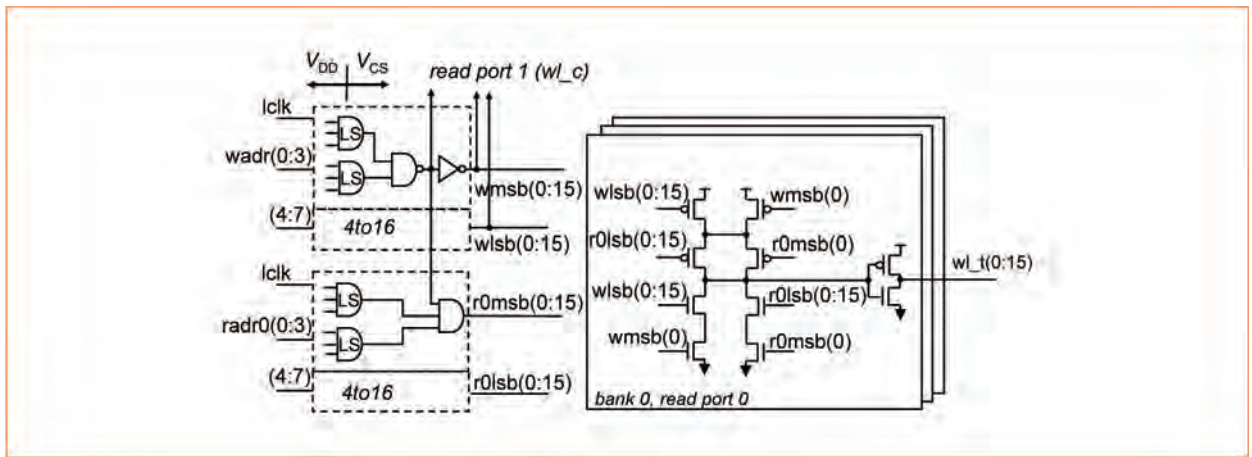


Figure 5

Write-over-read decode (©2011 IEEE).

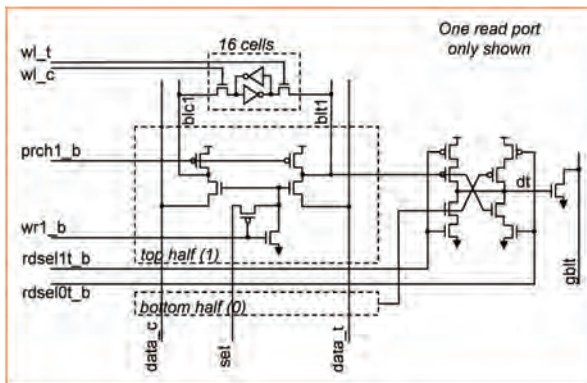


Figure 6

Local read and write circuit (©2011 IEEE).

bit line gblt. The explicit read select signal $rdsls[0, 1]_b$ has two functions: 1) select the read and block a write that could concurrently happen on the other bank 0 and 2) as a late power-gating signal, it blocks the propagation of the LBL onto the global bit line, where most of the array power is consumed due to the high signal count and capacitive load.

To write, the $wr1_b$ signal selects the row to be written (bank 1), and the set signal selects the column to be written. Both signals together ($wr1_b$, set) pull node w high initiating the write. One of the precharged bit lines $blt1$, $blc1$ is pulled down through Nwc/Nwt depending on the data polarity ($data_t$, $data_c$). With both word lines wl_t and wl_c being selected, the cell is differentially written. With the explicit column control, any combinations of bit/byte/way select can be implemented.

Summary

Key circuit techniques employed to build the POWER7 processor (the newest member of the POWER* processor family) have been presented. Superior on-chip memory densities are achieved by the use of eDRAM. The internal bandwidth requirements of the eight superscalar out-of-order quad-threaded cores are supported by innovative multiport register file and SRAM design concepts as double-pumped write ports and 6T cell-based multiport L1 caches. Chip frequencies over 4 GHz are achieved with precise clock generation and distribution. The DPLL assists multiple power modes to minimize power for a given workload. New levels of performance and scalability are achieved with the high-bandwidth on-chip I/O designs.

Acknowledgments

The POWER7 design has become a reality because of the dedication of the IBM processor design and manufacturing team around the globe.

*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 Sony Computer Entertainment Corporation in the United States, other countries, or both.

References

1. D. Wendel, R. Kalla, R. Cargoni, J. Clables, J. Friedrich, R. Frech, J. Kahle, B. Sinharoy, W. Starke, S. Taylor, S. Weitzel, S. G. Chu, S. Islam, and V. Zyuban, "The implementation of POWER7™: A highly parallel and scalable multi-core high-end server processor," in *Proc. IEEE ISSCC Dig. Tech. Papers*, 2010, pp. 102–103.
2. D. Wendel, R. Kalla, J. Warnock, R. Cargoni, S. G. Chu, J. G. Clables, D. Dreps, D. Hruscky, J. Friedrich, S. Islam, J. Kahle, J. Leenstra, G. Mittal, J. Paredes, J. Pille, P. J. Restle,

- B. Sinharoy, G. Smith, W. J. Starke, S. Taylor, J. Van Norstrand, S. Weitzel, P. G. Williams, and V. Zyuban, "A highly parallel, scalable, multi-core high end server processor," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 145–161, Jan. 2011.
3. Intel Corporation, *Intel QuickPath Architecture*, 2008. [Online]. Available: <http://www.intel.com/technology/quickpath/whitepaper.pdf>
 4. Intel Corporation, *Intel 7500/7510/7512 Scalable Memory Buffer*. [Online]. Available: <http://www.intel.com/Assets/PDF/datasheet/322824.pdf>
 5. S. S. Iyer, G. Freeman, C. Brodsky, A. I. Chou, D. Corliss, S. H. Jain, N. Lustig, V. McGahay, S. Narasimha, J. Norum, K. A. Nummy, P. Parries, S. Sankaran, C. D. Sheraw, P. R. Varanasi, G. Wang, M. E. Weybright, X. Yu, E. Crabbe, and P. Agnello, "45-nm silicon-on-insulator CMOS technology integrating embedded DRAM for high-performance server and ASIC applications," *IBM J. Res. & Dev.*, vol. 55, no. 3, pp. 5:1–5:14, May/Jun. 2011.
 6. J. Barth, W. R. Reohr, P. Parries, G. Fredeman, J. Golz, S. E. Schuster, R. E. Matick, H. Hunter, C. C. Tanner, J. Harig, K. Hoki, B. A. Khan, J. Griesemer, R. P. Havreluk, K. Yanagisawa, T. Kiriata, and S. S. Iyer, "A 500 MHz random cycle 1.5 ns, SOI embedded DRAM macro," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 86–95, Jan. 2008.
 7. J. Barth, D. Plass, E. Nelson, C. Hwang, G. Fredeman, M. Sperling, A. Mathews, W. Reohr, K. Nair, and N. Cao, "A 45 nm SOI embedded DRAM macro for Power7™ 32 MB on-chip L3 cache," in *Proc. IEEE ISSCC Dig. Tech. Papers*, 2010, pp. 342–343.
 8. G. Wang, D. Anand, N. Butt, A. Cestero, M. Chudzik, J. Ervin, S. Fang, G. Freeman, H. Ho, B. Khan, B. Kim, W. Kong, R. Krishnan, S. Krishnan, O. Kwon, J. Liu, K. McStay, E. Nelson, K. Nummy, P. Parries, J. Sim, R. Takalkar, A. Tessier, R. M. Todi, R. Malik, S. Stiffer, and S. S. Iyer, "Scaling deep trench based eDRAM on SOI to 32 nm and beyond," in *Proc. IEEE IEDM*, 2009, pp. 1–4.
 9. A. Allen, J. Desai, F. Verdico, F. Anderson, D. Mulvihill, and D. Kruger, "Dynamic frequency-switching clock system on a quad-core Itanium® processor," in *Proc. IEEE Solid State Circuits Conf.*, 2008, pp. 62–63, 63a.
 10. J. A. Tierno, A. V. Rylyakov, and D. J. Friedman, "A wide power supply range, wide tuning range, all static CMOS all digital PLL in 65 nm SOI," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 42–51, Jan. 2008.
 11. J. Pille, D. Wendel, O. Wagner, R. Sautter, W. Pentz, T. Froehnel, S. Buettner, O. Torreiter, M. Eckert, J. Paredes, D. Hruscecky, D. Ray, and M. Canada, "A 32 kB 2R/1W L1 data cache in 45 nm SOI technology for the Power7™ processor," in *Proc. IEEE ISSCC Dig. Tech. Papers*, 2010, pp. 344–345.

Received March 15, 2010; accepted for publication April 5, 2011

Dieter F. Wendel *IBM Deutschland Research and Development GmbH, Schoenaicherstrasse 220, Boeblingen 71032, Germany (WENDEL@de.ibm.com)*. Mr. Wendel received his B.S. degree in electrical engineering from the University of Wuerzburg, Germany, in 1981. He joined IBM the same year working in several areas and several locations in the United States and Germany. He is an IBM Distinguished Engineer. Recent assignments have been circuit lead for the Cell Broadband Engine** processor and for the IBM POWER7 processor.

John Barth *IBM Systems and Technology Group, Essex Junction, VT 05452 USA (jbarth@us.ibm.com)*. Mr. Barth received his B.S. degree in computer engineering from Northeastern University in 1987 and his M.S. degree in computer engineering from National Technology University in 1992. He joined IBM in 1987. His development efforts included standalone DRAM from 1987 to 1992, ASIC embedded DRAM from 1992 to 2004, and most recently SOI embedded DRAM targeted for the POWER7 on-chip L3 cache. His

memory designs can be found on Mars Pathfinder, IBM BlueGene* supercomputers, and POWER4*, POWER5, and POWER6 server L3 cache chips. He has coauthored numerous ISSCC papers and recently received an ISSCC Best Paper Award. He is an IEEE Senior Member, he served on the ISSCC Memory Subcommittee from 2000 to 2007, and currently serves on the Technical Program Committee for the VLSI Circuits Symposium.

Daniel M. Dreps *IBM Systems and Technology Group, Austin, TX 78758 USA (drepsdm@us.ibm.com)*. Mr. Dreps received his B.S.E.E. degree in 1983 from Michigan State University. He is a Distinguished Engineer working in the IBM Systems and Technology Group. During his IBM career, he has designed and developed transistor models, fiberoptic links, ASIC technology custom elements, and high-speed serial links for IBM servers. His interests currently focus on high-speed serial development and applications in the entire range of IBM servers. He has published multiple papers and holds more than 40 patents in broad areas of interconnect and server design.

Saiful Islam *IBM Systems and Technology Group, Austin, TX 78758 USA (saiful@us.ibm.com)*. Dr. Islam graduated from Bangladesh University of Engineering and Technology in 1987 and completed his M.S.E.E. and Ph.D. degrees at the University of Texas at Austin in 1990 and 1994, respectively. He worked with Advanced Micro Devices for almost nine years in the K5 and K7 microprocessor development team as a custom circuit designer. He joined IBM in 2003 and started working on the POWER6 project. Since then he has worked in the large register file array and complex custom circuit design area. Currently, he leads the custom register file team for future pSeries* servers.

Jürgen Pille *IBM Deutschland Research and Development GmbH, Schoenaicherstrasse 220, Boeblingen 71032, Germany (PILLEJ@de.ibm.com)*. Mr. Pille received his M.S. degree in microelectronics from the University of Hanover, Germany in 1990. He joined IBM at the IBM Deutschland Research and Development Lab, Boeblingen, Germany, in 1990 to work on advanced VLSI designs. Since then he has been working in several areas and several locations in United States and Germany. As a Senior Technical Staff Member, he is currently the pSeries server high-speed array lead.

Jose A. Tierno *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (tierno@us.ibm.com)*. Dr. Tierno received an engineering degree from Universidad de la Republica, Montevideo, and M.S. and Ph.D. degrees from the California Institute of Technology, in the area of self-timed digital circuit design. He has been a Research Staff Member at IBM Research since 1995 and has contributed to the design of magnetic read channels, high-speed serial links, digital phase-locked loops, and computer architecture.

Power optimization methodology for the IBM POWER7 microprocessor

V. Zyuban
J. Friedrich
C. J. Gonzalez
R. Rao
M. D. Brown
M. M. Ziegler
H. Jacobson
S. Islam
S. Chu
P. Kartschoke
G. Fiorenza
M. Boersma
J. A. Culp

Meeting the power budget of the 8 four-way simultaneous multithreading core IBM POWER7® microprocessor without compromising the aggressive performance targets presented a considerable challenge to the design team. Major innovations in the power modeling and power reduction methodologies have been introduced at all levels of the design, including microarchitecture, logic, circuits, postlayout tuning, and technology optimizations. In order to use effectively design resources available for power reduction, the team needed to understand precisely where the power was spent and the sensitivity to design parameters. A new power modeling methodology was deployed that allowed the team to evaluate the impact of design changes and various power reduction actions before sending them to the designers.

Introduction

Since the microprocessor industry first acknowledged power consumption as a primary design limitation, a number of power reduction techniques have been considered and successfully implemented in recent high-performance microprocessors [1]. Techniques such as clock gating, pulsed operation of latches, insertion of sleep transistors, use of high-threshold complementary metal-oxide semiconductor (CMOS) devices on noncritical paths, and partitioning of the chip into multiple clock and voltage domains have been extensively applied in microprocessors [2, 3]. When pervasively applied, these techniques reduce the power consumption in most areas of the chip, but the extent to which they are used may differ from unit to unit and from processor to processor. In addition to these common approaches, there are power reduction techniques that are unique to a particular architecture or implementation.

In this paper, we categorize the power reduction techniques employed by the IBM POWER7® microprocessor into microarchitecture, logic, and circuit levels, focusing on techniques that either are unique to POWER7 or required significant changes to the design methodology, allowing an increase in scope. For example, at the microarchitecture

level, we describe rearchitecting the register rename logic and issue queues and the introduction of embedded dynamic random-access memory (eDRAM) for data storage in the level-3 (L3) cache. At the logic level, this paper covers changes in the design flow that allowed the team to improve considerably the efficiency of clock gating, as well as innovative techniques to increase the effectiveness of the clock gating in arrays and register files. At the circuit level, this paper describes the double-pump operation of the WRITE ports of the two major register files and the innovative implementation of the register-file cell storage that allowed the increase in the number of bits needed to support four threads per core at a lower power cost than the traditional implementation. Even though the transistor tuning and the CMOS threshold swapping are considered a standard circuit-level power recovery, we improved the tuning flow to increase the return on investment.

The power reduction work at every level tends to compete for resources with the tuning of the microarchitecture to improve the performance and the circuit delay reduction to increase the maximum frequency. In order to use effectively design resources available for power reduction, the team needs to understand precisely where the power is spent and its sensitivity to various design options. The last section of this paper covers the new power modeling methodology deployed for POWER7 that allowed the team to evaluate the impact of design changes and potential power reduction actions before taking them to the designers.

Digital Object Identifier: 10.1147/JRD.2011.2110410

©Copyright 2011 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied by any means or distributed royalty free without further permission by computer-based and other information-service systems. Permission to republish any other portion of this paper must be obtained from the Editor.

0018-8646/11/\$5.00 © 2011 IBM

Microarchitectural improvements

The POWER7 team evaluated numerous architectural alternatives for power efficiency to arrive at the microarchitectural definitions that went into the implementation phase. We give five examples of innovations at the architectural level that allowed a significant reduction of the power.

The first example is replacing the general-purpose register (GPR), floating-point register (FPR), and vector register (VR) mapper used in previous IBM server processors [4] with a unified mapper. The virtual-to-physical register mappings for the architected state and the in-flight register state are stored in separate tables, whereas the in-flight mapper for the GPR, FPR, and VR are consolidated into one table. The new organization resulted in fewer latches, smaller area, and lower leakage power in the corresponding units.

The second example is a unified register-file issue queue that replaced the compacting latch-based queues in previous designs [4]. The use of the register-file-based structure takes considerably less alternating-current (ac) power since it eliminates the need to clock multiple latches to move instructions from the top to the bottom of the issue queue. Also, storing instruction bits in register-file cells rather than latches reduces the overall leakage power of the issue queues. However, the replacement of the compacting issue queue with register-file-based structures may potentially lead to some loss in the performance because of an extra cycle required to read operand indexes from register files compared with having them instantly available at the latch outputs. This was compensated by implementing a dispatch bypass and increasing the sizes of the issue queues, which achieved a lower overall power.

The third example is the mitigation of the power impact due to the increased architected register state in the simultaneous multithreading 4 (SMT4) mode. The GPR file has two instances that are coherent in the single thread and the SMT2. Each instance has READ ports for one fixed-point unit (FXU) and one load-store unit (LSU), although both instances have WRITE ports for all FXUs and LSUs. In order to limit the size of the register file needed to support four threads, the register-file instances are not coherent in the SMT4 mode, but rather, each register-file instance holds GPRs for only two threads. The issue queue is also split into two halves such that a given queue half-feeds the FXU and the LSU for a given thread pair. Instructions are assigned to a queue half at the dispatch time according to their thread ID. Thus, each issue queue has access to the GPR instance-holding registers for all instructions in the queue, eliminating the need for the operand crossbar switch and reducing the wire distance.

Fourth, the vector-and-scalar unit (VSU) implemented an innovative vector-register-file (VRF) architecture employing cell bit multiplexing to store bits from multiple

threads in one register-file cell. Each of the four instances of the VRF has six READ ports to feed three operands to each of the four double-precision floating-point units (FPUs). The VRF cell stores 2 bits that share the access circuitry, i.e., an even bit for threads 0 and 2, and an odd bit for threads 1 and 3, as shown in **Figure 1**. Two multiplexers that are internal to the VRF cell select between even and odd bits to connect to the READ access circuitry for each of the two groups of three READ ports [5]. This allows the POWER7 VSU to double the amount of register storage without doubling the power-expensive register-file access circuitry. The impact of the increased number of register-cell devices on the leakage power was mitigated by using superhigh V_t threshold devices for the cell and forcing transistor stacking as described in the section on circuit-level innovations.

Finally, in the memory system, a considerable power reduction was achieved by implementing a relatively small but fast 256-KB/core level-2 (L2) cache, backed up by a 4-MB/core L3 cache. The use of eDRAM arrays for the L3 data storage eliminated the static RAM (SRAM) cell leakage component [6]. The ac power of refreshing eDRAM cells is estimated as one fifth to one tenth of the leakage power through the equivalent number of densest SRAM cells, depending on the operating conditions. The size, the latency, and the bandwidth of L2 and L3 caches were balanced to achieve the minimum total power of the cache hierarchy.

Power reduction at the logic level

To improve the efficiency of clock gating, aggressive targets were set for the logic team using tables that convert microarchitectural events reported by a high-level processor model to the clocking cycles for latches and accesses to register files and arrays. With this approach, the clock gating targets can be derived for a wide range of workloads and processor utilization scenarios. When the logic of the processor was complete, the very high-speed integrated-circuit hardware description language (VHDL) was attributed with statements to extract the latch clocking events and the array access events. The extraction results were measured against the targets for a set of workloads. The quality of clock gating was further checked by a newly developed GateAlert tool [7].

In order to achieve the aggressive clock gating targets, the logic team developed new clock gating techniques, exploiting clock gating opportunities at a very fine level. In the FPUs, for example, parts of the data path are clock gated based on the instruction type, the result precision, and the operand values. These techniques allowed the FPU to gate latches 18% to 37% of cycles even under the peak performance [8].

Table 1 shows the VHDL-extracted unit-averaged latch clock gating for a power virus workload executing close to four instructions per cycle (IPC), and a low IPC workload with a high miss rate in the instruction cache.

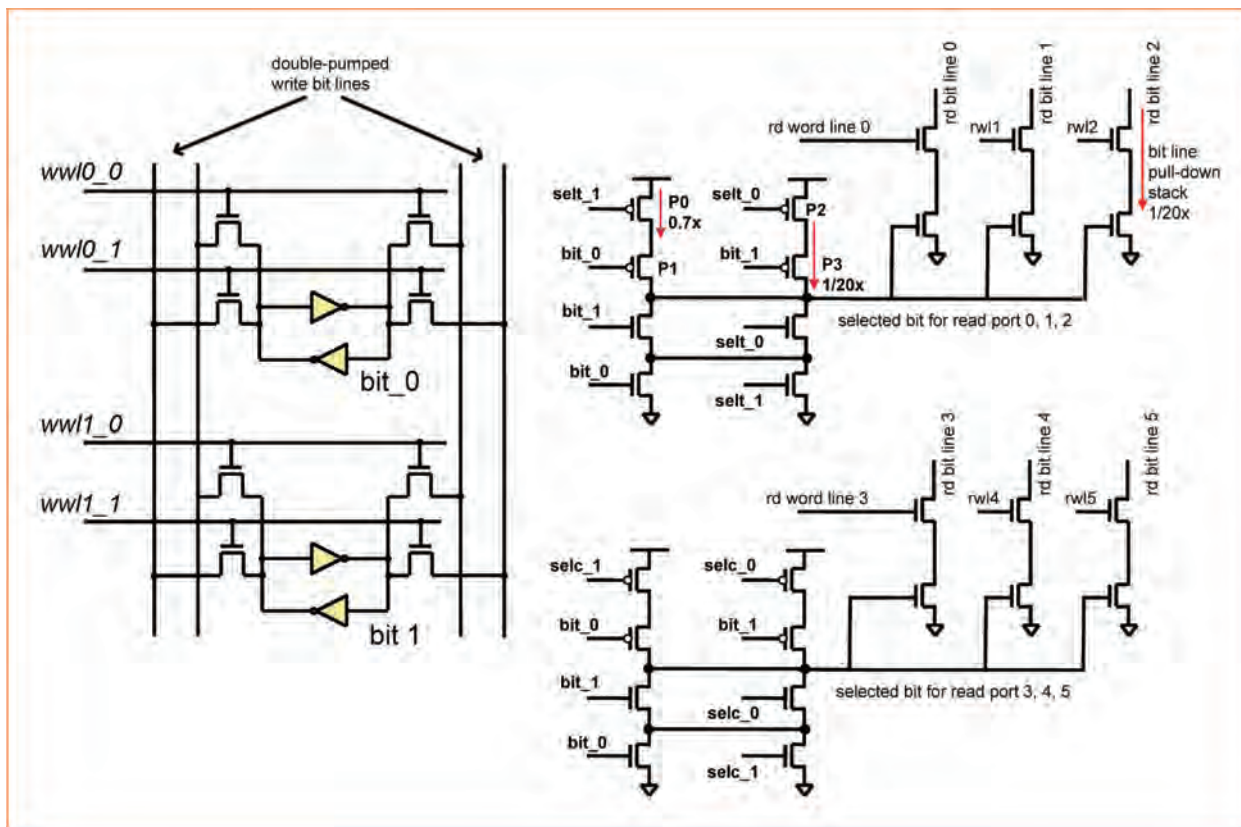


Figure 1

VRF cell capable of holding 2 bits.

The clock gating is implemented at the local clock buffer (LCB) level [9]. Latches connected to one LCB make up a clock gate domain and must use the same gating condition. Smaller clock gating domains typically present more opportunities for the clock gating at a cost of increasing the total number of LCBs. The tradeoff between the clock gating efficiency and the number of latch bits in clock gating domains was resolved on a per-domain basis. Typically, to justify the splitting of a clock gating domain below 16 bits per LCB, the clock gating of the involved latches would need to be improved by at least 20%. The bar for improvement in the clock gating was lower for latches that serve as data transition barriers that protect the downstream logic from switching.

The efficiency of the clock gating of array and register-file ports was improved by supporting multiple levels of gating. The first level of gating (called early gating) stops the clocking of the LCB driving the access to the port and achieves most of the power savings. However, it requires the earliest setup time for the gating signal. In many of the arrays, this level of savings can be achieved only for very

Table 1 Latch count and extracted unit-averaged latch clock gating.

Unit	Latch count	Power virus	Low IPC workload
DFU (decimal unit)	5K	0.83	0.97
FXU (fixed-point unit)	10K	0.88	0.94
IFU (instruction fetch unit)	32K	0.62	0.77
ISU (instruction sequencing)	40K	0.57	0.85
LSU (load store unit)	38K	0.62	0.81
PC (pervasive, debug, trace)	0.3K	0.52	0.57
VSU (vector and floating point)	26K	0.45	0.97

coarse gating conditions (e.g., no issue of any type to a unit). The second level of gating blocks the activation of the word line but does not stop power dissipation in the LCB of the corresponding access port and in one or more levels of the port decoder. Compared with the first level of gating, it allows for one or two additional levels of logic to compute the port gating conditions more precisely (e.g., including

Table 2 VRF and GPR: Design statistics and power reduction due to double pumping.

	<i>GPR</i>	<i>VRF</i>
Number of instances per core	2	4
Number of entries	112	172
Data width (including ECC bits)	72	74
Read ports (per instance)	4	6
Write ports (per instance)	4	4
Read, global bit-line switching, mW per core	3.1	11.0
Write, cell local inverters switching, mW per core	5.5	31.2
Leakage, cell inverters and bit line drivers, mW per core	21.4	30.2
Total power reduction per core, mW	30.0	72.4

the type of the instruction issued). The third level of gating intercepts the discharge of the global READ bit line. The additional setup time enables even more precise gating conditions (e.g., accounting for the data width required for the access). Finally, the last level of gating stops the switching of the outputs of the array [10].

In associative arrays with a late select, a targeted READ access mode was implemented to save power for accesses when the set is known at the beginning of the access cycle. The power is reduced by intercepting the access to columns that hold data of associative sets other than the requested one. This technique proved particularly useful in the late-select second-level cache, where the 128-byte line transfer happens in four sectors, accessed in the consecutive cycles. Three out of four accesses take advantage of the targeted access mode.

Power reduction at the circuit level

Implementing large multiport GPR and VRF files presents a considerable power challenge. Double pumping the WRITE ports allows doubling the WRITE throughput without increasing the number of cell access devices and the WRITE bit-line drivers. The power savings from double pumping includes the switching power in the cell local WRITE-data inverters that would otherwise be required because of the wiring track limitations and the leakage through the register-file access devices [pass-gate n-channel field-effect transistors (n-FETs) in the cells, cell WRITE-data inverters, and WRITE bit-line drivers]. Additional power benefits come from shorter WRITE and READ bit lines. **Table 2** gives design statistics for the GPR and the VRF, and a breakdown of the power savings achieved by double pumping WRITE under the power virus workload that issues approximately 1.6 double-precision vector multiply-adds per cycle and 1.6 vector loads per cycle. The total power reduction is more than 100 mW per core.

As described in the section on logic, the VRF stores 2 bits of data in one cell [5]. The power overhead consists

of a pair of bit multiplexers per cell, implemented as OR–AND–invert (OAI) multiplexers embedded in every cell, and decode circuitry driving two pairs of select lines across the array per row (selt0 and selt1, and selc0 and selc1), as shown in Figure 1. To reduce the leakage through the bit select OAI multiplexers and the bit-line pull-down n-FET stacks, all of the select lines (selt0, selt1, selc0, and selc1) for rows that are not accessed are set high. This forces the outputs of all cell OAI multiplexers to the ground, which puts all of the n-FETs in OAI multiplexers in a nonleaking state and forces stacking in the READ bit-line pull-down n-FETs (both n-FETs in the stack are off). The leakage of the OAI multiplexer depends on the contents of the cell. For example, if bit_0 is at zero and bit_1 is at one, then the leakage scaling factor for p-channel FET (p-FET) P0 is 0.7 (because of the V_t drop across p-FET P1), and the leakage scaling factor for p-FETs P2 and P3 is about 1/20 (since both p-FETs in the stack are off). On the average, only one of the four p-FETs (P0, P1, P2, and P3) in the OAI is in a leaking state; therefore, the OAI multiplexers contribute only 2.3 mW of leakage in each VRF instance. However, since the outputs of the OAI multiplexers are at zero, the stacking in the READ bit-line pull-down n-FETs reduces the average leakage through the READ bit-line pull-down stacks by 5.4 mW per VRF instance. The cell multiplexer select decoders have a leakage overhead of approximately 5 mW. Thus, because of the circuit-level innovation to reduce the leakage, the overall leakage cost of the cell bit multiplexing is only 2 mW per VRF instance. The power benefit of the VRF cell bit multiplexing with respect to doubling the number of array instances to get the same cell count is estimated as 180 mW of leakage per core.

Other power-saving techniques implemented in POWER7 arrays and register files include the floating WRITE and READ bit lines, the local power gating of decoders, the engineering of n- or p-FET stacks to reduce leakage, and the precharge of the global bit line at the beginning of

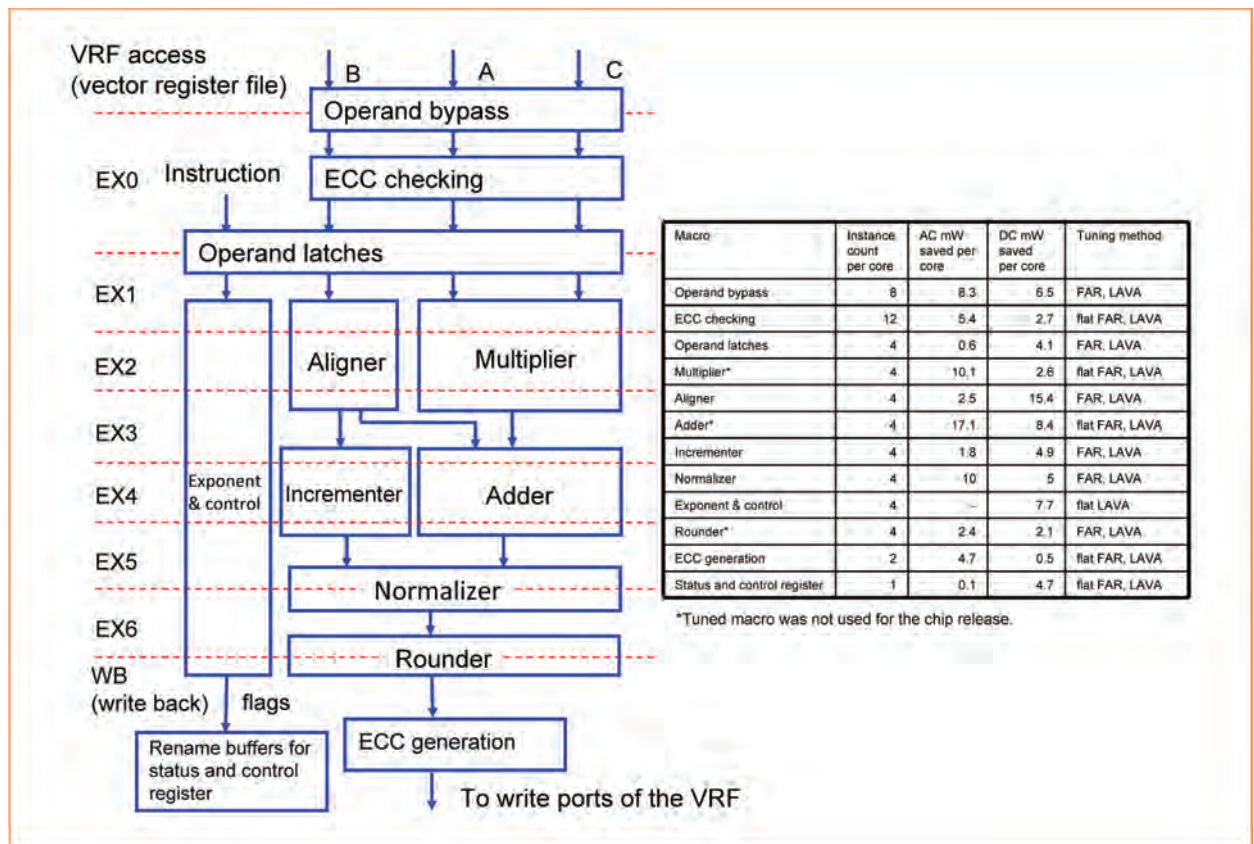


Figure 2

Power reduction in the floating-point pipeline due to macro tuning.

the READ access, rather than at the end. A larger list of circuit-level improvements includes the double-bit latch, where 2 bits of the latch share a local clock inverter, granular LCBs to match precisely the power level of the local clock drivers to the extracted capacitive load of the local clock wires, and the latches driven by each LCB.

Macro tuning and postlayout power recovery

Macro tuning for low power, employing the EinsTuner tool [11], was extensively used to recover power during the initial design and after completing the physical layout of macros. To maximize the power recovery at a fixed design resource allocated for the power tuning effort, we used a novel tuning assessment methodology [12] that evaluates the power recovery potential based on the number of tunable objects and the number of paths with positive timing slack. The total macro tuning potential was estimated to be approximately 500 mW per core. Out of half a thousand eligible macros, only 100 macros were selected for tuning, leading to the power recovery of approximately 400 mW per core.

The mechanics of the tuning process was different for custom and synthesized macros and also dependent on the switching activity in the macros. Custom macros with a high switching activity were tuned using the EinsTuner free-area recovery mode for device width optimization followed by the Leakage Avoidance and Analysis (LAVA) mode for device threshold optimization. Macros with a low switching activity and synthesized logic macros ran EinsTuner only in the LAVA V_t optimization mode. The use of size-parameterized library elements, available in three levels of CMOS thresholds, allows the tuner to reduce the device sizes or swap the MOSFET threshold without disturbing the macro layout or wiring. Furthermore, a new flat tuning methodology was applied to a few select macros. The flat tuning methodology involves flattening a hierarchical custom macro to library cells, which exposes more tunable objects to EinsTuner.

Figure 2 shows tuning results for a double-precision floating-point pipeline [8], which is instantiated four times in the vector unit (VSU). The reduction in the dynamic power (ac) is stated for a VSU-intensive workload that

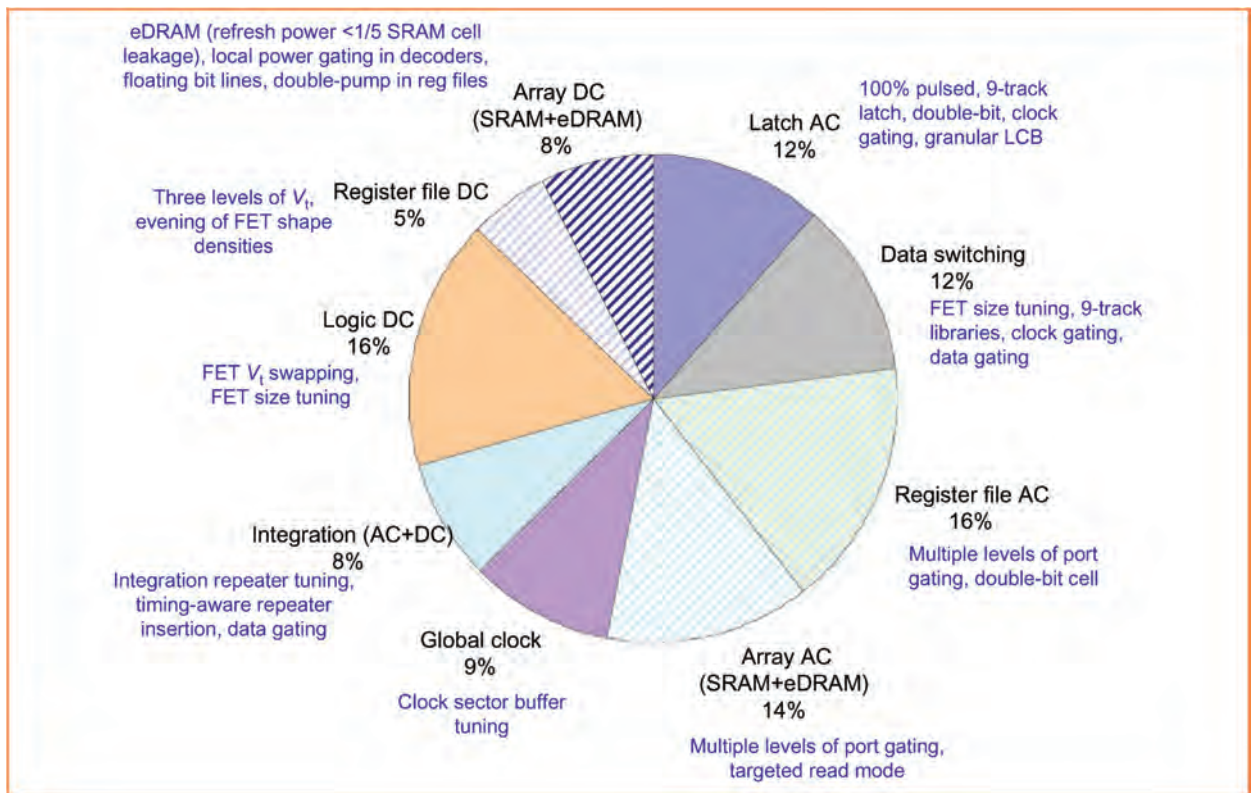


Figure 3

Chiplet power breakdown by type and the corresponding circuit-level power reduction techniques.

completes 1.5 vector multiply-adds per cycle. Notice that, after tuning, a macro needs to go through the design sign-off process, which includes checks for logical and physical correctness, timing, signal integrity, reliability, testability, and yield. Passing these checks for some of the macros required an additional design effort, which is the reason why some of the tuned macros could not be included in the chip release (marked by asterisk).

The tuning process was also applied to interconnect repeaters, which are a significant source of chip leakage. A new simultaneous-switching-noise timing-aware buffer environment was introduced that leverages the three V_t MOSFET offerings, as well as fine-grain buffer sizing to reduce the buffer power by 25% by trading the positive timing slack for the higher threshold and the smaller buffer size. This new environment iteratively took into account any simultaneous-switching-noise impact due to buffer changes and ultimately allowed more buffer power savings without having an impact on the timing target.

The ability to tune independently logic macros and interconnect repeaters presents an interesting optimization problem. The timing slack on a path between macros can be

used to downsize transistors inside either the macros or the interconnection repeaters. The optimal partitioning of the slack between macros and repeaters depends on a number of factors, such as logic fan-out, switching activity, and number of tunable objects inside macros. However, from the design-effort perspective, giving priority to the tuning of the integration repeaters typically showed a higher return on investment.

The POWER7 team also partnered with the IBM process development team to minimize the power. A new process was used to reduce the across-chip variation that leads to the higher chip leakage by measuring the transistor shape density and providing feedback to the design team. Semiautomatic routines were then developed to allow designers to smooth out circuit density variations by inserting dummy circuits in low density areas such as nest logic and global wiring areas so that those regions would more closely match the dense circuitry regions found in the processor cores and caches. The updated final density data was generated and then used by the postprocessing fill routines typically used to improve planarization to further smooth the circuit density on wafers.

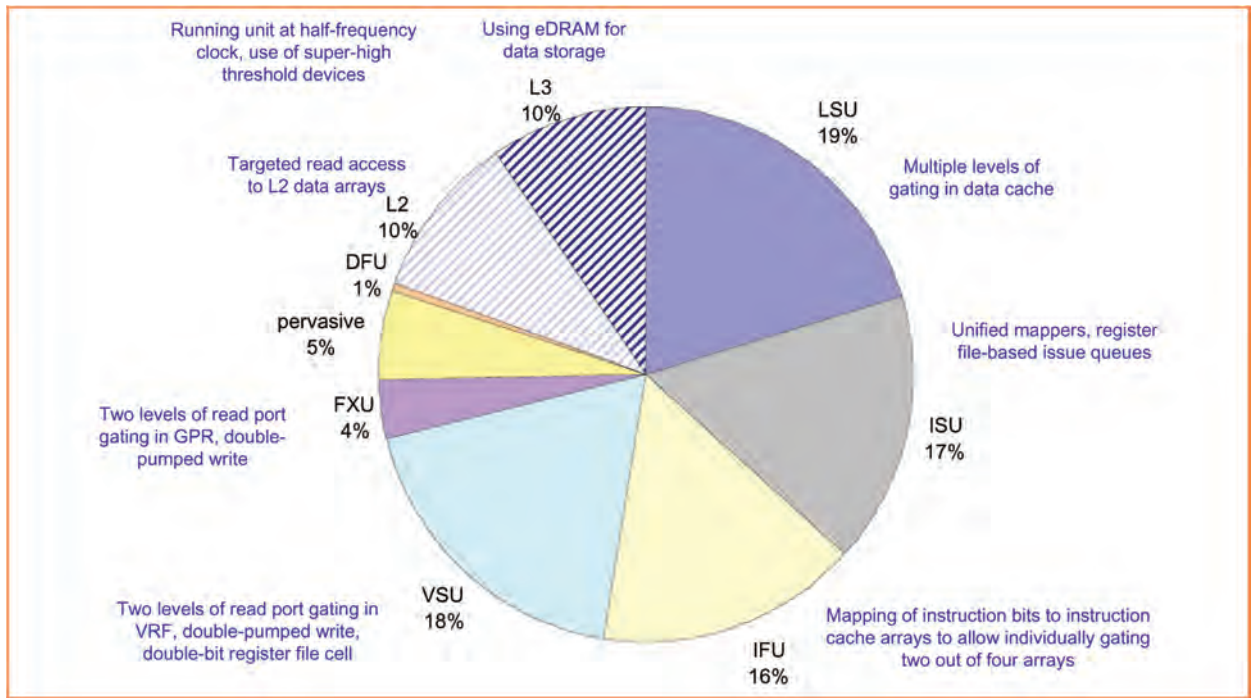


Figure 4

Chiplet power breakdown by unit and the corresponding microarchitectural power reduction techniques.

Power modeling and tracking methodology

POWER7 created a new power modeling methodology that allowed the team to evaluate the impact of design changes and various power reduction actions. The newly introduced tools included a power database and a power calculator that tracked the circuit and logic simulation results against power contract expectations, a new microarchitecture level tool that gathered activity statistics for key workloads from the processor performance model, and a new tool for predicting the power of register files and arrays based on an analytical model. We also introduced a new version of the circuit-level simulator capable of extracting device leakage probabilities and stacking factor, measuring the ac power dissipated in every mode of the arrays and register files, and automatically creating a parameterized power rule called the *power abstract* that allowed the computing of the power for any access and data switching patterns. Designers were required to run the power extraction tools for the design sign-off, and the created power abstracts were promoted to the design database. Rather than storing the power from a simulation of fixed input vectors at one power, voltage, and temperature (PVT) point, the new system created an activity-parameterized power rule in terms of effective capacitance and leaking gate width. This rule could be then paired with VHDL

simulations to calculate the power across PVT points and workloads.

Conclusion

The power breakdown of the POWER7 chiplet by type is shown in **Figure 3**. The chiplet includes core, L2, and L3 caches, and contributes two thirds of the total chip power. The remaining one third is dissipated in the input/output (I/O) and the nest, and includes power bus, symmetric multiprocessing buses, memory and I/O controllers, and chip pervasive logic. Figure 3 gives examples of the circuit-level power reduction techniques that were used in POWER7. **Figure 4** also shows the chiplet power breakdown by unit, giving examples of microarchitecture-level power reduction techniques used in the corresponding units.

The state-of-the-art power reduction flow has reached a level where there are no longer any simple pervasive techniques that would allow considerable power recovery across multiple units at a low implementation cost. In fact, the power has been distributed over many areas, and no single area has presented a particularly easy power recovery opportunity. Thus, there has been no single design technique to reduce the power across the whole chip. Achieving new levels in the power reduction to enable the continued growth in the processor performance requires a global focus and considerable resource investment.

Acknowledgments

The authors would like to thank the work of the power tools team for creating the new power modeling infrastructure. In particular, we would like to thank D. Thomas, B. Agarwal, I. Govett, N. Dhanwada, D. Hathaway, S. Neely, D. Winston, and P. Kamdar. Also, the authors would like to thank the individual contributions of the following: unit circuit/power leads J. Stuecheli, G. Wiedemeier, B. Burky, A. Bianchi, M. Genden, M. Erle, J. Leenstra, M. Delaney, B. Philhower, A. Butts, J. Summers, and J. Nahidi; circuit/power leads of the array and register-file teams B. Truong, G. Mittal, R. Sautter, J. Atwal, S. Barnes, J. Noack, U. Brandt, J. Pille, S. Salewski, D. Plass, J. Barth, and C. Hwang; GateAlert team E. Arbel, C. Eisner, and O. Rokhlenko; EinsTuner and BufOpt team T. Mitchell, E. Cho, and J. Neves; the developers of the new chip fill process S. Runyon and J. Preston; and the rest of the POWER7 team and S. Kosonocky. This material is based on the work supported by the Defense Advanced Research Projects Agency under its Agreement HR0011-07-9-0002.

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

References

1. S. Naffziger, "High-performance processors in a power-limited world," in *VLSI Symp. Tech. Dig.*, Jun. 2006, pp. 93–97.
2. S. Rusu, S. Tam, H. Muljono, J. Stinson, D. Ayers, J. Chang, R. Varada, M. Ratta, and S. Kottapalli, "High A 45 nm 8-core enterprise Xeon processor," in *Proc. ISSCC*, Feb. 2009, pp. 56–57.
3. J. Friedrich, B. McCredie, N. James, B. Huott, B. Curran, E. Fluhr, G. Mittal, E. Chan, Y. Chan, D. Plass, S. Chu, H. Le, L. Clark, J. Ripley, S. Taylor, J. Dilullo, and M. Lanzerotti, "Design of the POWER6 microprocessor," in *Proc. IEEE ISSCC Dig. Tech. Papers*, San Francisco, CA, Feb. 11–15, 2007, pp. 96–97.
4. T. N. Buti, R. G. McDonald, Z. Khwaja, A. Ambekar, H. Q. Le, W. E. Burky, and B. Williams, "Organization and Implementation of the register-renaming mapper for out-of-order IBM POWER4 processors," *IBM J. Res. Dev.*, vol. 49, no. 1, pp. 167–188, Jan. 2005.
5. D. Wendel, R. Kalla, R. Cargoni, J. Clabes, J. Friedrich, J. Kahle, B. Sinharoy, W. Starke, S. Taylor, S. Weitzel, S. G. Chu, S. Islam, and V. Zyuban, "The implementation of POWER7: A highly parallel and scalable multi-core high-end server processor," in *Proc. IEEE ISSCC Dig. Tech. Papers*, 2010, pp. 102–103.
6. J. Barth, D. Plass, E. Nelson, C. Hwang, G. Fredeman, M. Sperling, A. Mathews, W. Reohr, K. Nair, and N. Cao, "A 45 nm SOI embedded DRAM macro for the POWER7 processor 32 MByte on-chip L3 cache," in *Proc. ISSCC*, 2010, pp. 342–343, Paper 19.1.
7. E. Arbel, C. Eisner, and O. Rokhlenko, "Resurrecting infeasible clockgating functions," in *Proc. 46th Annu. DAC*, 2009, pp. 160–165.
8. J. Preiss, M. Boersma, and S. M. Mueller, "Advanced clockgating schemes for fused-multiply-add-type floating-point units," in *Proc. 19th IEEE Symp. ARITH*, Jun. 2009, pp. 48–56.
9. J. Warnock, L. Sigal, D. Wendel, K. P. Muller, J. Friedrich, V. Zyuban, E. Cannon, and A. J. Kleinosowski, "POWER7 local clocking and clocked storage elements," in *Proc. IEEE ISSCC Dig. Tech. Papers*, 2010, pp. 178–179.
10. J. Pille, D. Wendel, O. Wagner, R. Sautter, W. Pent, T. Froehnel, S. Buettner, O. Torreiter, M. Eckert, J. Paredes, D. Hrusecky,

D. Ray, and M. Canada, "A 32 kB 2R/1W L1 data cache in 45 nm SOI technology for the POWER7 processor," in *Proc. IEEE ISSCC Dig. Tech. Papers*, 2010, pp. 344–345.

11. A. R. Conn, P. K. Coulman, R. A. Haring, G. L. Morrill, and C. Visweswariah, "Optimization of custom MOS circuits by transistor sizing," in *Proc. IEEE/ACM ICCAD Dig. Tech. Papers.*, 1996, pp. 174–180.
12. M. M. Ziegler, V. Zyuban, G. D. Gristede, M. Vratonjic, and J. Friedrich, "The opportunity cost of low power design: A case study in circuit tuning," in *Proc. 14th ACM/IEEE ISLPED*, 2009, pp. 133–138.

Received January 8, 2010; accepted for publication October 18, 2010

Victor Zyuban IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (zyuban@us.ibm.com). Dr. Zyuban received an M.S. degree from Moscow Institute of Physics and Technology and a Ph.D. degree in computer science and engineering from the University of Notre Dame. He joined IBM in 2000. He was the core power lead on the POWER7 processor and is currently leading the power modeling and optimization on one of IBM's future designs. He is also manager of a low-power circuits group at the IBM T. J. Watson Research Center.

Joshua Friedrich IBM Systems and Technology Group, Austin, TX 78758 USA (friedrich@us.ibm.com). Mr. Friedrich was the circuit design leader for the POWER6* microprocessor memory subsystem, and after the completion of its first-pass design, he led its circuit characterization, optimizing both frequency and yield. He led the design of several core units in the POWER5 and has been instrumental in power reduction efforts as power lead for POWER7. Mr. Friedrich is currently working as chip circuit lead and has been a driving force behind the tool, design methodology, and technology team for one of IBM's future designs.

Christopher J. Gonzalez IBM Systems and Technology Group, Technology Development, Yorktown Heights, NY 10598 USA (jgonzalz@us.ibm.com). Mr. Gonzalez received a B.S. degree from Rutgers University and an M.S. degree from Columbia University. He has been with IBM since 2001 working on custom circuit and unit design. For the POWER7 processor, he led the circuit design of a core execution unit and is currently the core power lead for one of IBM's future designs.

Rahul Rao IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (raorahul@us.ibm.com). Dr. Rao received his Ph.D. degree from the University of Michigan in 2004, for this thesis on low-power VLSI design. He is a Research Staff Member at the T. J. Watson Research Center and was actively involved in the design of the POWER7 processor. Additionally, his work has focused on characterization of variations and device degradation, and design of robust high-performance memories.

Mary D. Brown IBM Systems and Technology Group, Technology Development, Austin, TX 78758 USA (marybrown@us.ibm.com). Dr. Brown received the B.S. degree in computer science from Florida State University, the M.S. degree in computer science and engineering from the University of Michigan, and the Ph.D. degree in computer engineering from the University of Texas. She has been at IBM since 2005, working on microarchitecture, logic, power, and performance of the POWER7 instruction sequencing unit.

Matthew M. Ziegler IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (zieglerm@us.ibm.com). Dr. Ziegler received a Ph.D. degree in electrical engineering from the University of Virginia in 2004. After

graduation he joined the IBM T. J. Watson Research Center as a Research Staff Member. His contributions to POWER7 included designing the floating-point multiplier, driving the tuning power reduction effort, and custom design methodology development. His research currently focuses on developing new design methodologies for design productivity and low power.

Hans Jacobson *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (hansj@us.ibm.com).* Dr. Jacobson received his M.S. degree from University of Lulea, Sweden, and his Ph.D. degree from University of Utah. He has been with IBM Research since 2001 where he has worked on power modeling, design, and verification of several IBM microprocessors and test chips. He is currently power lead in research toward an exascale processor. His research interests include advanced pipelining and clocking, microarchitectures and circuits, and power modeling.

Saiful Islam *IBM Systems and Technology Group, Technology Development, Austin, TX 78758 USA (saiful@us.ibm.com).* Dr. Islam graduated in electrical engineering from Bangladesh in 1987 and received M.S.E.E. and Ph.D. degrees from the University of Texas at Austin in 1990 and 1994, respectively. He worked with AMD on K5 and K7 as a circuit designer starting in 1995. In 2003 he joined the IBM POWER6 team. Since then he has been working on register files and complex circuits. Currently, he is leading the custom register file team for one of IBM's future designs.

Sam Chu *IBM Systems and Technology Group, Technology Development, Austin, TX 78758 USA (chusg@us.ibm.com).* Mr. Chu received his master's degree from Syracuse University and joined IBM in 1984 where he has been working in different processor design areas. He is an IBM Senior Technical Staff Member. He is currently the technical lead of register file designs, supporting the IBM Z- and P-processor families.

Paul Kartschoke *IBM Systems and Technology Group, Technology Development, Essex Junction, VT 05452 USA (pkartsch@us.ibm.com).* Mr. Kartschoke is a Senior Engineer in the microprocessor design group in Burlington, Vermont. He received his B.S. degree in electrical engineering from Rutgers University in 1990 and his M.S. degree in electrical engineering from the University of Virginia in 1992. He joined IBM in 1992 and works on timing takedown and methodology for IBM CMOS microprocessors.

Giovanni Fiorenza *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (gfiorenz@us.ibm.com).* Dr. Fiorenza received his Ph.D. degree in physics from Stevens Institute of Technology in 1992. He joined IBM in 1984 and has been working at the IBM T. J. Watson Research Center since 1996 as a circuit and physical designer for the POWER4 through POWER7 microprocessors. He is currently working in the same capacity on one of IBM's future designs.

Maarten Boersma *IBM Systems and Technology Group, Technology Development, Schoenaicherstrasse 220, Boeblingen, Germany 71032 (MBOERSMA@de.ibm.com).* Mr. Boersma received an M.Sc. degree in electrical engineering from the University of Twente, the Netherlands. Since 2005 he has worked for IBM in Boeblingen, Germany as logic designer for high performance floating-point units. He contributed to the double-precision floating-point unit for the IB PowerXCell* 8i synergistic processing element and co-designed the floating-point unit for the POWER7 processor. His focus is on high-frequency, power-efficient design and on formal verification techniques.

James A. Culp *IBM Systems and Technology Group, Technology Development, Hopewell Junction, NY 12533 USA (jaculp@us.ibm.com).* Mr. Culp received his bachelor's and master's degrees in electrical engineering from Drexel University. His research focus has been on modeling and mitigating circuit limited yield in advanced technology nodes.

Adaptive energy-management features of the IBM POWER7 chip

M. Floyd
M. Ware
K. Rajamani
T. Gloekler
B. Brock
P. Bose

A. Buyuktosunoglu
J. C. Rubio
B. Schubert
B. Spruth
J. A. Tierno
L. Pesantez

The IBM POWER7® processor implements several new adaptive power-management techniques that, in concert with the EnergyScale™ firmware, allow it to proactively take advantage of variations in workload, environmental conditions, and overall system utilization to meet customer-directed power and performance goals. These features build on the support and the capabilities provided by its predecessor, i.e., the IBM POWER6™ processor. Among these are per-core frequency scaling with available autonomous frequency controls, per-chip automated voltage slewing, power-consumption estimation, soft power capping, and hardware instrumentation assist.

Introduction

Managing the power-and-performance tradeoff of a running computer system is a complex problem. In the IBM POWER7* processor, there are a multitude of low-level controls for the behavior of the hardware, but these also have impacts on the performance, which vary based on the type of workload and the combination of workloads being processed at a given time. Although there is no one-size-fits-all policy that can be implemented, EnergyScale* employs an adaptive approach to the problem in the form of a joint hardware, firmware, and software solution [1]. A dedicated off-chip microcontroller, coupled with policy guidance from the customer and feedback from the POWER Hypervisor* and operating systems that are running on the machine, determines modes of operation and the best power/performance tradeoff to implement during runtime to meet customer goals while achieving the best possible performance. Like its predecessor, POWER7 employs the more traditional “dynamic” power-saving techniques such as clock gating circuits when the clock is not needed, runtime scaling of the frequency and the voltage to adjust to varying utilization, and various sensors to measure the environment and workloads under which the

chip is operating [2]. Several adaptive power-management features were added to POWER7 to augment these capabilities, and these are described in this paper.

Functional overview

The POWER7 chip features eight processor core chiplets, each consisting of the processor core with its associated level 2 (L2) and level 3 (L3) caches. These eight core chiplets are all fed by a common external power source, separately controllable from the rest of the chip. For modularity and design reuse, each core chiplet runs asynchronously to the others and with respect to the SMP interconnect fabric. For maximum power, thermal, and yield concerns, the static and embedded dynamic RAM (eDRAM) circuits are powered with a separate slightly higher voltage than the rest of the logic circuits in the core chiplet; thus, two core chiplet voltages must be managed by the system independently of the voltage supplied to the rest of the chip.

Figure 1 provides an overview of the POWER7 adaptive energy-management infrastructure described in greater detail in the remainder of this paper.

The external EnergyScale microcontroller has direct access to both chip and chiplet power-management controls and sensors through the internal control bus. However, many key functions are also provided by the chip power-management controller (PMC), which controls core voltages and serves as an on-chip proxy for the EnergyScale controller during certain power-management

Digital Object Identifier: 10.1147/JRD.2011.2114250

©Copyright 2011 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied by any means or distributed royalty free without further permission by computer-based and other information-service systems. Permission to republish any other portion of this paper must be obtained from the Editor.

0018-8646/11/\$5.00 © 2011 IBM

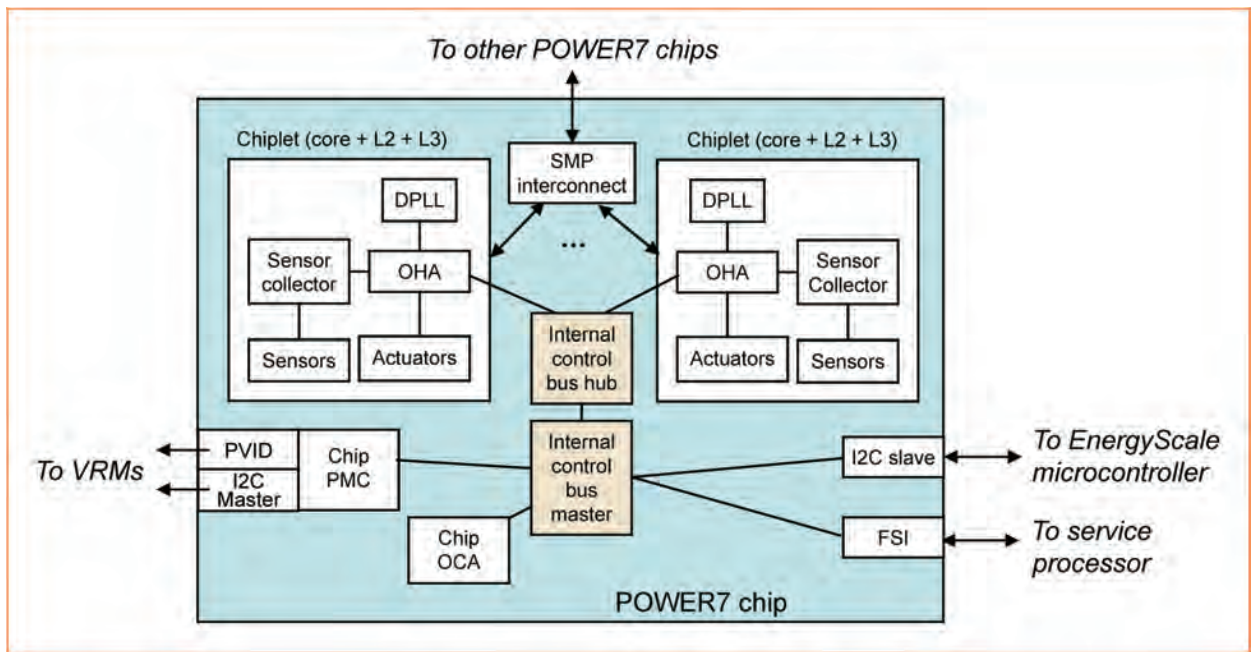


Figure 1

Structural overview of the on-chip power-management infrastructure.

protocols. In the core chiplets, the on-chip hardware assist (OHA) function coordinates the frequency management through the digital phase-locked loop (DPLL) and autonomous actuators. High-frequency sensor data is also sampled and reduced by the OHA and periodically collected by the on-chip communications assist (OCA) unit for eventual streaming access by the EnergyScale controller.

Per-core asynchronous frequency scaling capability

Since the cores in previous POWER* systems ran synchronously to the symmetric multiprocessing (SMP) interconnect, the EnergyScale firmware had to slew the frequency of the reference clock for the entire system in order to change the frequency of the processor cores. This system-level process was relatively slow and affected every core and cache in the system, making it less useful and suboptimal for adaptive power-management algorithms. In addition, this method often constrained the EnergyScale firmware since the attached support chips (e.g., input/output controllers) also ran synchronously to this SMP interconnect but could operate only within certain frequency ranges, differing based on the type and the version of the support chips present in a given system. In addition, multiple processor cores on a chip traditionally run off a shared analog phase-locked loop (APLL), which is a design that is not amenable to frequency slewing. To work around this

problem, some systems have incorporated dynamically selectable $M : N$ fractional frequency operation to downstream circuits either internal or external to the APLL. Per-core frequency scaling was introduced in the Advanced Micro Devices (AMD) Opteron** processor [3]. The introduction of DPLLs for per-core scaling in POWER7 allows much finer grain scaling capability (28-MHz step sizes) with fewer restrictions on desirable frequencies. With the scale-up to eight processors on a chip, running all cores at the same frequency or even from a subset of independently configurable fractional frequencies off the same reference would limit the effectiveness of the EnergyScale firmware to match the power efficiency to the demand (utilization and workload type) running on each core. Additionally, for a fractional frequency chiplet solution, transactions across the interface must be carefully handled by programmable ratio synchronization logic, which adds complication and latency into the performance-critical processor-memory SMP interface. POWER7 leverages the asynchronous core chiplet design by supporting a 50%–110% slew from a nominal frequency while minimizing the latency by using a synchronous-WRITE asynchronous-READ array queuing structure across the chiplet interface. To provide this flexibility, a new circuit was employed to allow EnergyScale to control the frequency independently and dynamically during runtime on a per-core-chiplet basis.

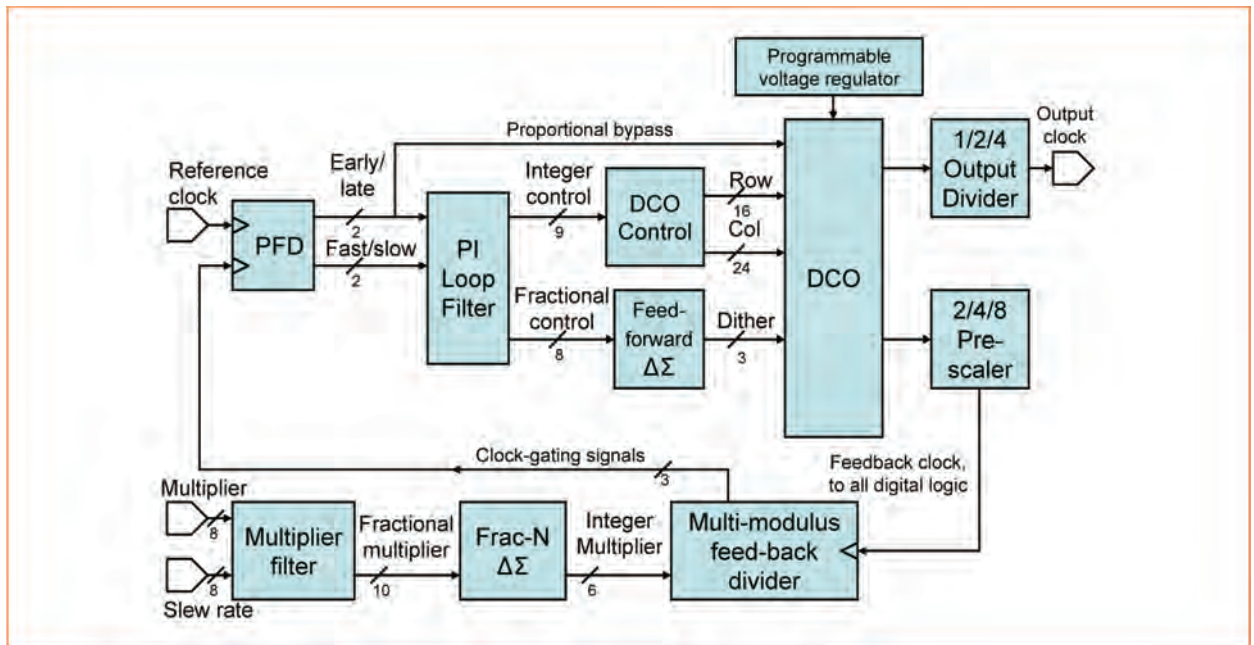


Figure 2

Top-level block diagram of the DPLL.

DPLL

The per-core chiplet variable frequency generator is built around a fractional- N DPLL. **Figure 2** shows a top-level block diagram of the circuit. The key components of the DPLL-based generator include a bang-bang self-timed phase-and-frequency detector (PFD), a digital loop filter, a digitally controlled oscillator (DCO), prescalers, a multimodulus feedback divider together with a delta-sigma modulator, and a multiplier filter.

The output frequency of the DPLL can be dynamically changed by controlling the modulus of the feedback divider. The function of the multiplier filter is to generate a sequence of modulus values resulting in a controlled programmable frequency slew rate. When a new target frequency multiplier *Mult_{in}* and frequency slew rate *Mult_{slew}* are programmed into the filter, a linear sequence of multipliers *Mult_{out}* is generated by incrementing or decrementing the internal *Mult_{acc}* register until its value matches the new target multiplier. The slew rate of the *Mult_{out}* multiplier is controlled by dividing down the clock applied to the multiplier filter, according to the *Mult_{slew}* value.

The bang-bang operation of the modified PFD on POWER7 is similar to that reported in [4]. For this modified PFD design, cycle slips are detected for downstream use. Note that, when a cycle slip occurs, a second edge of one of the clocks arrives without an intervening edge of the other clock. In this case, within the PFD, the second edge

of the fast clock arrives when the corresponding edge detector latch is still set, as *Reset* can be asserted only after both clock edges have arrived. The corresponding *Faster* signal will then be set, indicating which clock is running faster. Because the *Faster* signals are asserted every time a cycle slip happens, they are asserted with a frequency that is proportional to the difference in frequency between the two clocks. Once the DPLL is locked, the two clock frequencies are the same, and the *Faster* signals are automatically de-asserted.

Measurements of the controlled-frequency transients taken from a POWER7 core chiplet DPLL, for three different settings of the slew rate, are shown in **Figure 3**. **Figure 4** shows the final piece of the transient, where three distinct regions are clearly identifiable, i.e., frequency acquisition, phase acquisition, and phase lock. In the frequency-acquisition region, the generator output follows the frequency multiplier ramp. Every time a cycle slip occurs between the reference and the feedback clock, the output frequency is changed by approximately 30 MHz. Once the frequency multiplier is stable and the generator output is close to the target frequency, cycle slips stop occurring. In this phase-acquisition region, the final phase and frequency are attained using the integrator in the DPLL filter. The proper selection of DPLL filter constants ensures that no overshoot or undershoot in cycle time is present in the transient. The combination of the controlled-frequency

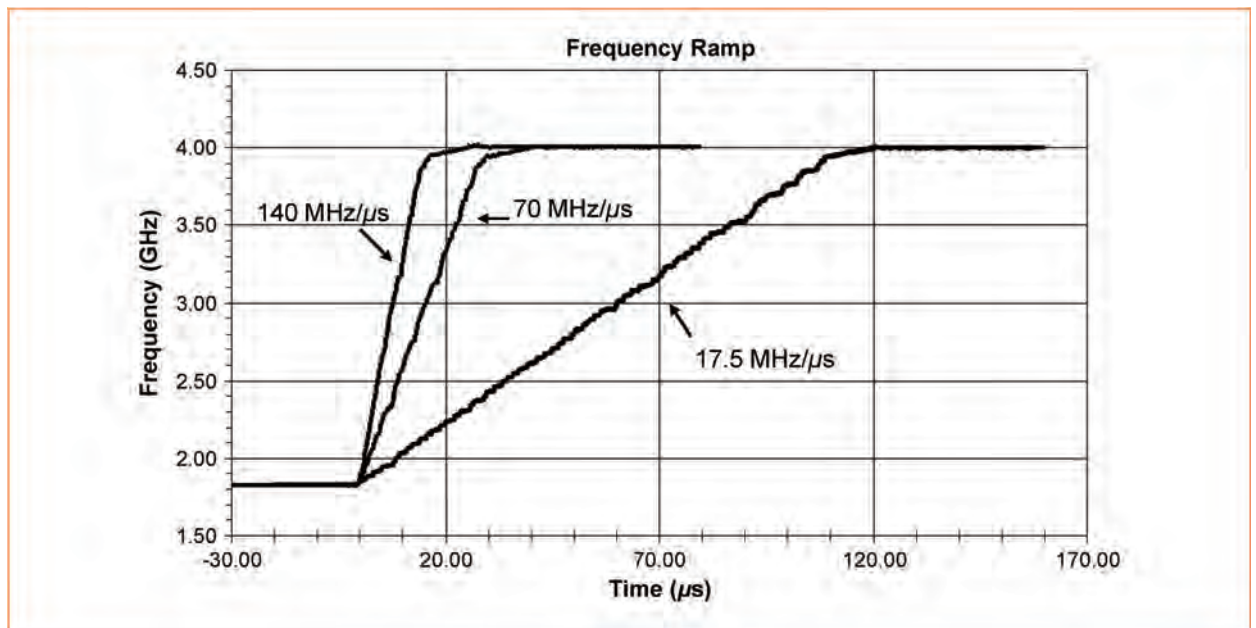


Figure 3

Measurements of the controlled-frequency transients taken from a POWER7 core chiplet.

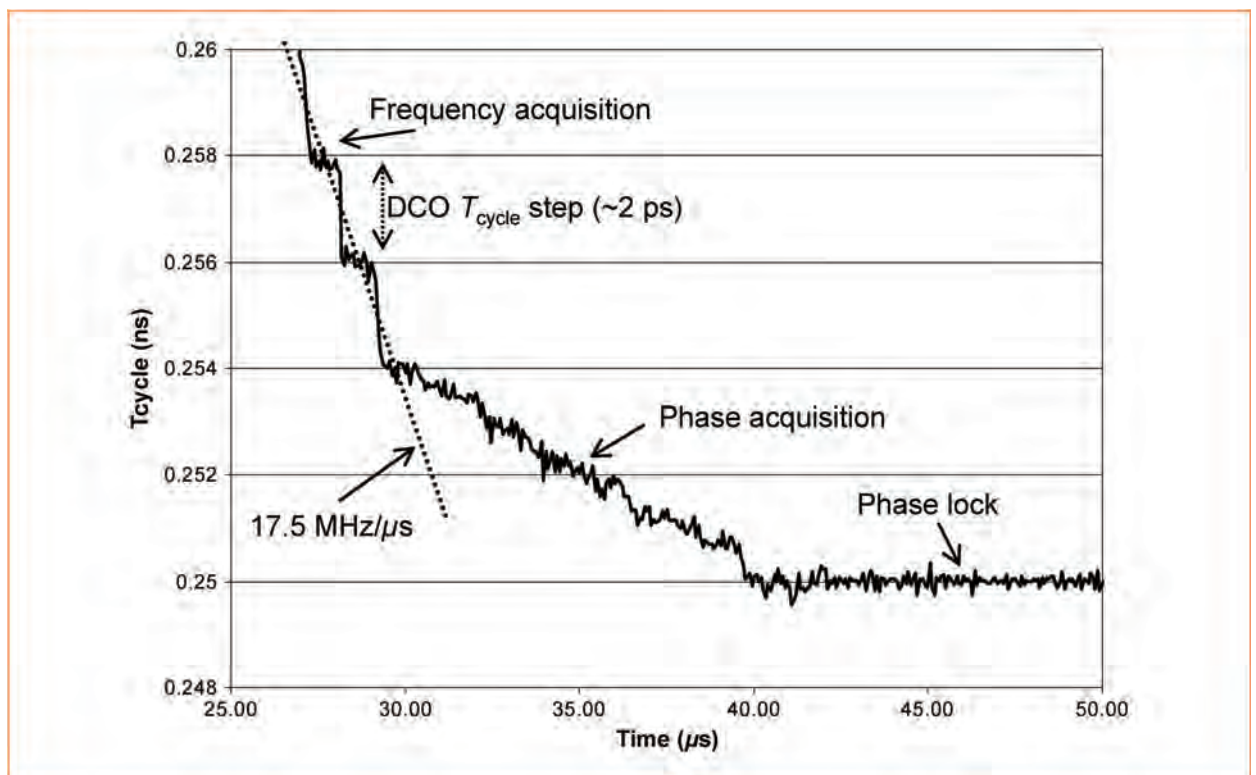


Figure 4

Detailed measured transient showing cycle time T_{cycle} against time at the end of a frequency ramp.

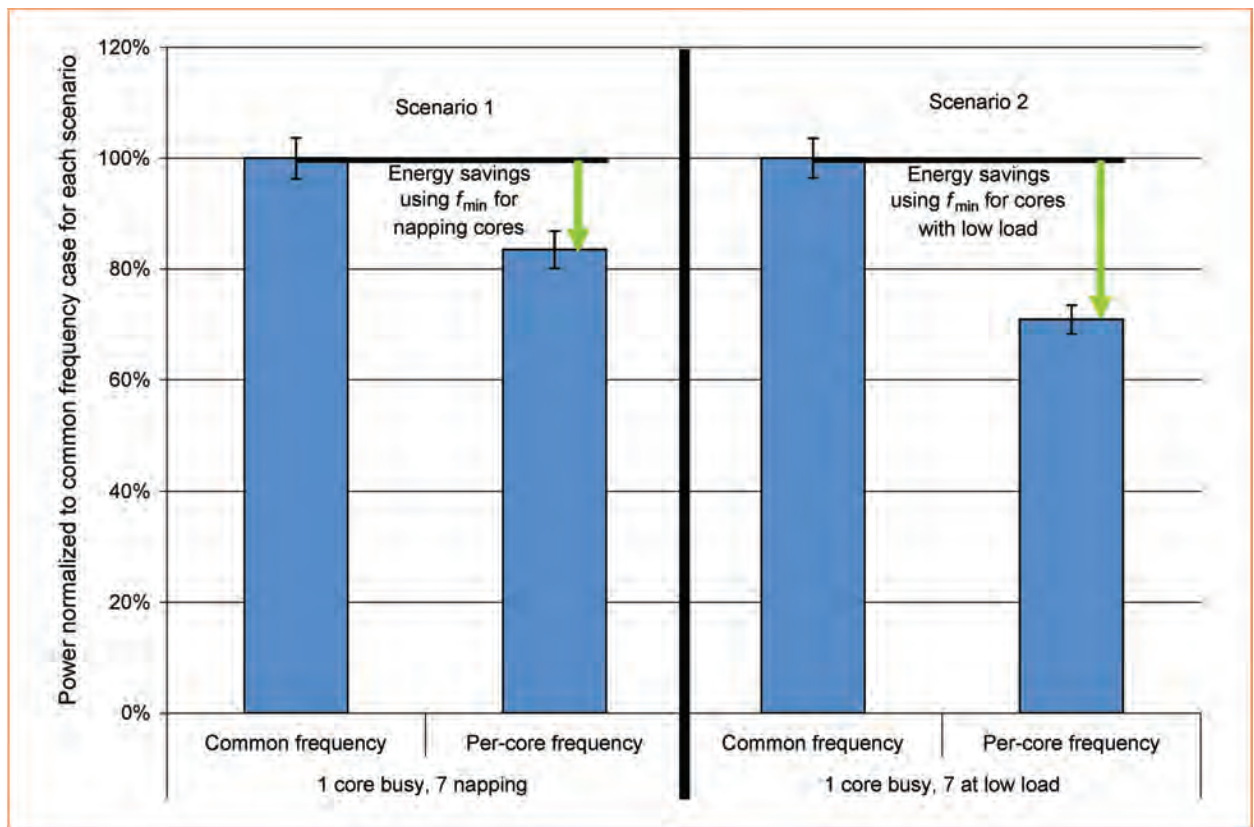


Figure 5

Better energy reduction from the DPLL with per-core frequency scaling.

slew rate and the absence of short cycles allows POWER7 to use this DPLL-based variable frequency generator to dynamically adjust the frequency of the core chiplet while the core is a fully operational executing code.

The DCO has a measured tuning range of 800 MHz to 12 GHz. The DCO output is divided by two to improve the duty cycle of the clock provided to the processor core logic. The root-mean-square period jitter is 1 ps at 5 GHz and 6 ps at 1 GHz. The DPLL has an area of $200 \mu\text{m} \times 350 \mu\text{m}$, half of which is occupied by the voltage regulator for the DCO. The DPLL slews the frequency without pausing instruction execution. It can slew a frequency range of more than 2 GHz in less than $50 \mu\text{s}$. Consequently, POWER7 achieves a dramatic improvement in both the slew rate and the frequency range compared with the IBM POWER6* processor.

Two scenarios where the per-core frequency scaling capability with the DPLL provides increased energy efficiency are shown in **Figure 5** as an illustrative example. Scenario 1 is when a single core in a chip is busy at 100% load and the seven others are napping. Scenario 2 is when a single core is similarly busy while the other seven are also

running but with the demands of their workloads met by running at the minimum allowed frequency. While both scenarios show energy reductions with the per-core scaling, the higher benefits for scenario 2 show the DPLL's unique value for energy-efficiency improvements when all cores are active.

Coherence interconnect command rate

Incorporating asynchronous support into the SMP interconnect design was not trivial. For a maximum performance including both SMP link and memory bandwidth, the interconnect fabric does not use a tagged coherence protocol with handshakes like many other asynchronous core microprocessor designs. Coherence transactions are conducted using a nonblocking broadcast-based transport mechanism, where responses to broadcast events are collected, arbitrated, and aggressively reordered by the local snoopers to ensure a coherent operation at the maximum throughput.

To prevent overrunning buffers, the allowed command rate for these coherence operations must be limited such

that the capacity of the asynchronous buffers of the slowest core chiplet in the system is not exceeded. For this reason, a programmable coherence interconnect command rate (CICR) creates a lower limit on the level to which EnergyScale can reduce frequency in the system, since it imposes an operational frequency floor. However, reducing the CICR too much will penalize the performance by reducing the available SMP and the memory bandwidth available to every coherence requester in the system. Therefore, EnergyScale algorithms must carefully balance the CICR setting with the minimum frequency that it allows for the processor cores in the system. To assist in this analysis, the POWER7 hardware provides instrumentation in the form of programmable utilization “watermarks” that allow the firmware to observe the actual utilization of the SMP interconnect fabric during runtime. This is accomplished in the hardware by counting how often the amount of coherence activity during a specified time interval exceeds a certain threshold for two programmable threshold values. This capability allows the EnergyScale firmware to “learn” or characterize the coherence bandwidth currently needed. To react to changing workloads on the system and customer policy changes during runtime, the hardware supports changing this CICR value dynamically while the system is still running via a special control sequence similar to that used for concurrent processor node upgrade or repair.

Processor idle states

POWER7 supports two (nap and sleep) of the four idle states that are described in the IBM Power Architecture* [5]. These two states represent the middle of the power efficiency versus latency tradeoff. The power reduction is determined by which regions of the chip are “turned off” and how they are turned off. This is directly related to the latency, which is determined by how much state must be purged during entry and must be restored in the process of waking up from that idle state. Analysis showed that doze, i.e., the lightest idle state, would have no measurable latency or power benefit since POWER7 implemented a lower latency nap. Winkle, the heaviest idle state, requires some type of power-off capability combined with some style of power-on assist and was not implemented in POWER7.

Nap

The nap idle state simply quiesces the instruction fetch and execution, and completely turns off all clocks to the execution engines in the processor core (which, in practice, affects only the latches not already turned off naturally by local clock gating when the execution pipelines go idle). Nap on POWER7 keeps the chiplet caches (L1, L2, and L3) coherent and does not purge them, unlike nap on POWER6, whose sole supported idle state lies roughly halfway in between the POWER7 implementation of nap and sleep. Like its predecessor, POWER7 nap causes a thread switch,

resulting in a thread-specific program state loss, but the Hypervisor state is maintained. The latency of nap on POWER7 is typically approximately 2 μ s.

Heavy nap

POWER7 offers a mode to the EnergyScale firmware that causes nap to additionally slew down the frequency of the processor core while it is in the nap state. Note that, since the caches are still running, the frequency cannot drop below that supported by the CICR. Upon wakeup, instruction execution commences immediately, while the DPLL starts to restore the chiplet back to its target operational frequency. Reducing the frequency in the nap mode may affect the performance of the other cores on the chip that may be using this still-running shared L3 cache to hold lateral castouts; therefore, it is enabled only when the customer chooses a power saving policy.

Sleep

Sleep is a more aggressive idle mode than nap, in which it clocks off the entire core chiplet including the caches and even its own clock source (DPLL). For this reason, to maintain coherency, the L1, L2, and L3 caches must be purged to memory before entering this state and all translation lookaside buffers invalidated. Upon wakeup, the DPLL must be restarted, and the L3 cache’s eDRAM must be powered up and sequenced back to the functional operation. Sleep latency, as observed by an application, is typically less than 1 ms but may exceed 50 ms under rare worst case conditions. The sleep latency is dominated by the L3-cache purge on entry (whose timing varies with the number of dirty lines and their placement in the physical memory) and the eDRAM re-enablement process upon wakeup. Note that sleep requires that the timebase facility be restored by the Hypervisor firmware.

Heavy-sleep state

When all eight core chiplets on the chip enter the sleep state, POWER7 can be configured to drop them to the retention voltage, i.e., a level at which latches and arrays safely retain their state for extended periods of time. The latency of heavy sleep is typically 10–20 ms (depending on system configuration) and may also exceed 50 ms in a rare worst case. The additional latency for heavy sleep is due to the ramp down of the power planes to the retention voltage and then back to the operating voltage automatically by the hardware on wakeup.

Figure 6 shows the relative power reductions for nap and sleep as measured on a small sample set of POWER7 processors during the bring-up laboratory phase. For these measurements, the frequency ranged from $F_{\min} = (0.5 * F_{\text{nom}})$ to $F_{\max} = (1.1 * F_{\text{nom}})$, where the nominal core frequency F_{nom} was approximately 3.5 GHz. $V_{\min}$ and $V_{\max}$ refer to the set of voltages corresponding to

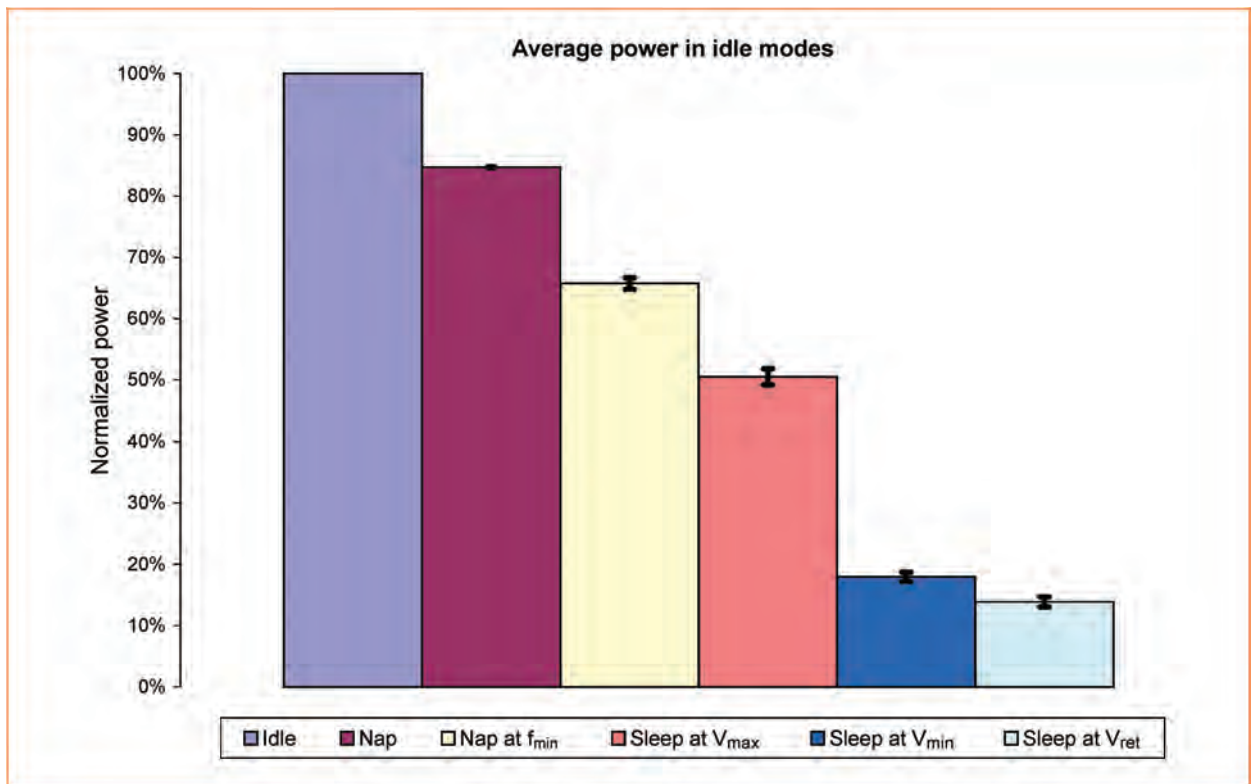


Figure 6

Power reduction with the POWER7 processor idle states.

those frequencies. Note that the benefit of the retention voltage in heavy sleep is larger when system constraints (e.g., CICR) prohibit the frequency and, therefore, the voltage from dropping all the way down to the chip's V_{min} .

Automated voltage slewing controls

To support an idle state such as heavy sleep, an external controller cannot respond quickly enough to support wakeup latencies in the required sub-10-ms time scale. Therefore, POWER7 integrates the voltage control into the chip for the chiplet array voltage V_{CS} and chiplet logic voltage V_{DD} . The off-chip power-management microcontroller does not control the targeted core chiplet voltages directly but instead makes voltage change requests through the POWER7 chip. The chip hardware, in turn, appropriately selects between the idle and operational target voltages. However, the dynamic voltage changes using standard voltage regulator modules (VRMs) cannot be made as a step function. The previous EnergyScale firmware spent a significant amount of its time sequencing voltage changes in order to elicit a smooth stable VRM response and avoid voltage transients during the transition. The POWER7 chip hardware automates this

function in the form of a sequencing engine, freeing up the compute power on the microcontroller for more advanced power-management algorithms. Programmable step sizes and delays for both core chiplet voltage rails allow the firmware to finely tune the slope of each voltage change. A change in the target voltage triggers the automated sequencer to step toward the new voltage level. The sequencer coordinates stepping by slewing V_{dd} and V_{CS} at different rates. After the voltage targets are both reached, the sequencer will wait a minimum programmable time before processing the next voltage target change request to ensure that the core chiplet has adjusted to the new voltage and is in a stable state before the next voltage change commences.

Low-end and midrange systems employing the POWER7 chip contain VRMs external to the chip, directly sourcing the two voltage rails to the core chiplet. They are controlled by an industry-standard 8-bit parallel interface [parallel voltage identifier (PVID)], which is a linear control code representing 6.25-mV increments. This technique is fast because it is limited only by the response time of the VRMs, which sample the PVID input in the megahertz range. High-end systems with greater power delivery and

redundancy requirements use power supplies that are actively managed by their own microcontroller to support more sophisticated techniques such as current sharing and failover to provide a more robust power supply. In these systems, the sheer number of processors and resulting scarcity of the available connector and board wiring make PVID interfaces impractical at the system level, and thus, the POWER7 chip instead provides a serial control interface based on the industry-standard Inter-Integrated Circuit (I2C) protocol. This interface is used to communicate the new voltage targets, relying on the power supplies' embedded controller to do the stepping. Because of the latency and the overhead of the I2C protocol, this technique is significantly slower than the PVID; thus, voltage adjustments take longer. However, since the power supply handles the stepping function internally, a single large voltage change request can be sent and serviced in less than 10 ms over the I2C interface, i.e., a latency allowing heavy sleep to be enabled in high-end systems.

Power delivery on high-end systems is organized as pairs of independent supplies sourcing common voltage rails using a current-sharing arrangement. The POWER7 serial voltage interface provides transparent hardware-controlled failover if one supply of the pair appears to have failed or becomes unresponsive to voltage change requests. POWER7 automatically discontinues the use of the questionable supply, controlling the system voltage through the remaining good supply. This allows the system firmware to attempt recovery or request replacement of the failed supply without the customer experiencing a system outage.

Power proxy

Effective power management in a microprocessor requires runtime measurement of the power. However, the measurement of real calibrated power consumption in the hardware can be a difficult and complex task. For example, physically measuring the power might require periodically stalling the processor for proper calibration. Isolating power consumption on a per-chiplet (combination of a single core with its L2 and L3 caches) basis is simply not possible externally to the chip when multiple chiplets share the same power grid, and reliable internal power measurement circuits were not available for POWER7. In the absence of the ability to do the actual measurement, a technique was developed to provide an estimate or "proxy" for the amount of power being consumed. Previous research [6] has examined the concept of utilizing the existing hardware performance counter data, collected by the software, to estimate the core or chip power consumption. However, choosing the right (and minimal) set of activity monitors and corresponding weights to estimate the power at high accuracy has not been published in such prior work. In terms of server-class processor designs, the Intel chip code named Tukwila is reported [7] to have a mechanism to choose

global (voltage and frequency) settings for the chip, based on the monitoring of activity levels across various regions within the chip. This facility allows maximizing the chip frequency for a detected level of activity. In that design, there is no notion of the accurate estimation of the power on a per-core or per-chiplet basis for the purposes of dynamic power shifting and management across on-chip and off-chip resources. AMD has very recently referred to an on-chip power monitoring and control facility [8] in one of its 32-nm x86-64 chips. This design is reported to use an activity counter-driven per-core power estimation facility. However, specific implementation concepts and detailed measurement-based performance (i.e., accuracy analysis) data have not been reported.

POWER7 implements a hardware mechanism in the form of a power proxy by using a specially architected programmable weighted counter-based architecture that monitors carefully selected activities and forms an aggregate value. Events are selected with an understanding of the microarchitecture of the processor, such that we can capture those that maximally correlate with the active-power consumption. The activity events A_i in the processor core, L2 cache, and L3 cache are each multiplied by a programmable weight factor W_i and then added together, with a constant offset for empirical curve fit, into a single composite activity value representing average power over a sampling interval, where $P_{act} = \sum(W_i * A_i) + C$. The estimation is performed across a programmable time-scale interval as small as a few microseconds. The event power P_{act} is then adjusted for active power not associated with the available activity events (such as clock grid power and other frequency-dependent events) by multiplying the average frequency over the interval by a final constant to generate a chiplet *power proxy* value, where $P_{proxy} = P_{act} + (K * F_{avg})$. By selectively weighing the different events and constants relative to one another based on the empirical postsilicon correlation work, the amount of the active power consumed by the chiplet can be estimated with an acceptable level of accuracy. The power-management firmware then adjusts the hardware activity-based power proxy for the effects of leakage, temperature, and voltage to estimate the total power of the chiplet. This power proxy is largely deterministic in nature, meaning that a given workload should roughly produce the same power-proxy "signature" every time it runs, since it is derived from signals that logically activate as the program executes. Of course, the normal variation in the performance occurs due to the different activity in the cache and the memory subsystem due to SMP effects of other workloads running; however, the power proxy will track the difference in the energy being accordingly consumed.

At the presilicon stage, the POWER7 methodology relied on systematic linear-regression-based formalism in conjunction with the designer intuition and experience to

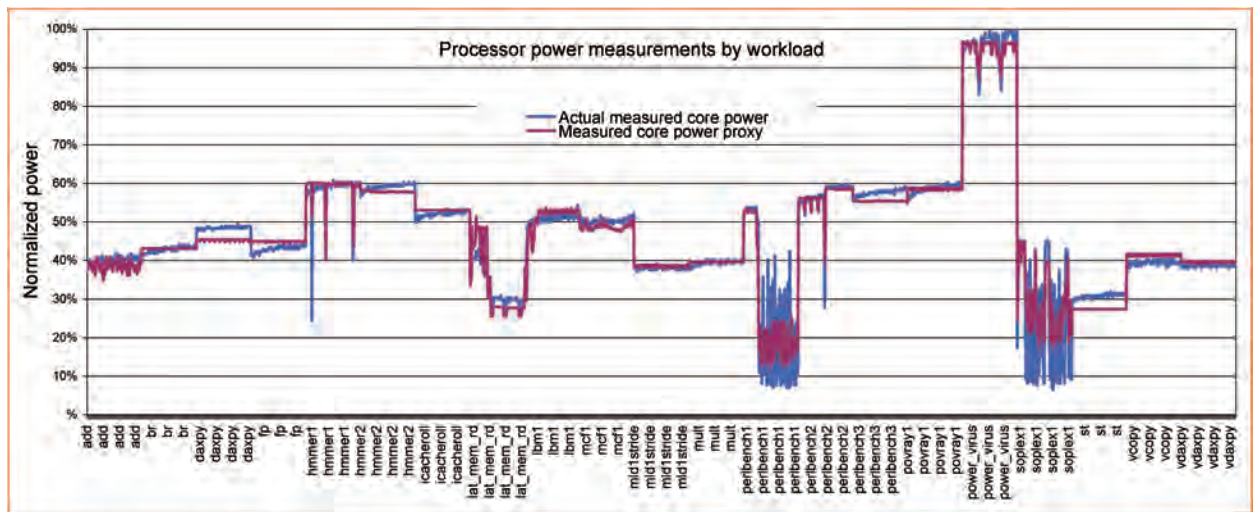


Figure 7

Empirically measured power versus estimated proxy values on the POWER7 chip hardware. The y-axis shows power data normalized to the workload representing the peak power.

choose the starting set of events and programmability. Since the hardware implementation is limited due to the physical silicon area, a technique had to be employed, which allowed the exploration of a solution that did not necessarily match the global optimum provided by traditional linear programming tools. For this reason, an IBM internal genetic algorithm programming tool called SNAP was utilized to further optimize the design under hardware implementation constraints. The genetic algorithm was configured to find the best least-square-type curve fit from all the possible linear equations that could be formed with the available hardware mechanism. This allowed several iterative changes to tune and tweak the design in order to achieve the best results. The final design measures close to 50 activity counts that include events such as instructions dispatched; instructions completed; execution register-file accesses; execution pipeline issue types; instruction fetch unit activity; load-store unit cache, translation buffer, and prefetch activities; and L2- and L3-cache READS/Writes. The weight registers are architected to be wide enough to capture relative contributions of different events. There is support in the architecture for providing negative weights for a small number of counters in order to correct for the effect of the overlap (double counting) of events through the set of counters selected. The design team found this necessary since events in a processor are hardly ever mutually exclusive; for example, for an instruction to complete, it must be fetched and dispatched and must usually access several register files and caches. A key challenge in the activity selection, of course, is to minimize such overlap and to identify the

fundamental (pseudoindependent) positive correlators of the power. Meeting this challenge successfully meant the minimal use of negative weights and an affordable number of hardware counters and weight bits. After some effort, this led to an accurate yet flexible power-proxy architecture with reasonable cost.

The postsilicon measurement-based weight calibration follows a similar systematic methodology as in the presilicon stage. First, sample points are collected for a series of targeted benchmarks such that all the power-proxy events are represented. Each sample point contains a power measurement (for the core and the L2 and L3 caches) and a count for each activity event. For the duration of the workloads, sample points are taken at a rate of a certain time granularity per sample. Once the data are gathered and postprocessed, the genetic algorithm can be run to determine all the power-proxy settings to provide an optimal curve fit.

The empirical data in **Figure 7** shows results of the power-proxy run against a diverse range of sample workloads and microbenchmarks collected using a real hardware. Empirical results demonstrate that the power-proxy estimates are very close to the actual power consumption in most cases. In our initial attempts to calibrate against the hardware, 91% of the samples fell within a $\pm 10\%$ relative error and 73% of the samples fell within a $\pm 5\%$ error. It is also interesting to note that our empirical experiments have shown that the variation in the data patterns being processed by some workloads alone can generate up to a 6% change in the power consumption for the same workload.

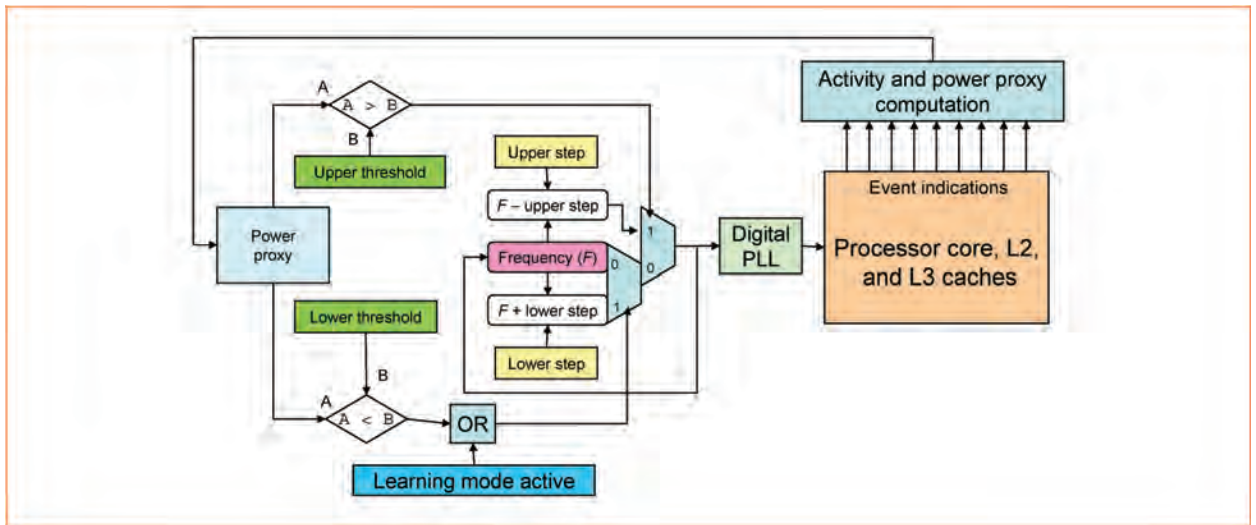


Figure 8

Feedback loop for soft-power capping.

Memory-controller power proxy

The POWER7 on-chip memory controllers also support weight-programmable event-based power proxies for the memory subsystem designed to drive local power control state machines in the memory controller to augment traditional performance-centric memory access scheduling with the power awareness. These are enhanced techniques that are more effective than those first explored in the POWER6 design.

Autonomous frequency control

The per-core asynchronous frequency scaling capability (as described earlier) allows POWER7 to implement several autonomous frequency control mechanisms. Autonomous frequency controls move processor core chiplet frequencies within a defined range in response to operating conditions, taking advantage of the fine-grained low-latency frequency control provided by the per-core DPLLs. The upper end of the frequency range is typically the nominal operating frequency for the current voltage, and the lower end of the range varies depending on the power-management policy and system-level timing constraints. Given the common voltage planes shared by all cores in a POWER7 chip, per-core autonomous frequency scaling can be an effective way to manage the power in a virtualized environment where interpartition constraints may preclude lowering common voltages. These autonomous capabilities offer better power efficiency and improved performance in power-constrained environments.

Low-activity detect

The POWER7 low-activity detect (LAD) mechanism manages core chiplet frequencies in response to workload changes and is activated on intervals as short as a few microseconds. The LAD is triggered by changes in per-core utilization, derived from a qualified instructions completed per cycle (IPC) measurement, lowering frequencies temporarily as long as the triggering metric is below a programmable threshold. This mechanism permits very rapid frequency scaling to exploit low-activity periods that are too short for the firmware or the system software to detect. For example, busy-wait polling loops can be modified to include simple code sequences that cause significant drops in the IPC during periods of unproductive polling. This leads to an immediate power saving due to circuit-level clock gating and can also trigger the LAD for even further power savings through autonomous frequency scaling. Software control loops executing on the EnergyScale microcontroller are constantly monitoring effective average frequencies of processor cores in relation to nominal frequency targets. In some power-management policies, the control loops would also lower the maximum frequency and associated voltages suitably after having observed that the effective average frequency is lower than the target as a result of LAD actions. Unlike this hardware-driven mechanism, similar approaches in the industry have been entirely software driven [9, 10], and some additionally require machine learning [11].

Soft-power capping and power shifting

Using these described capabilities, the EnergyScale microcontroller determines how to optimize the performance

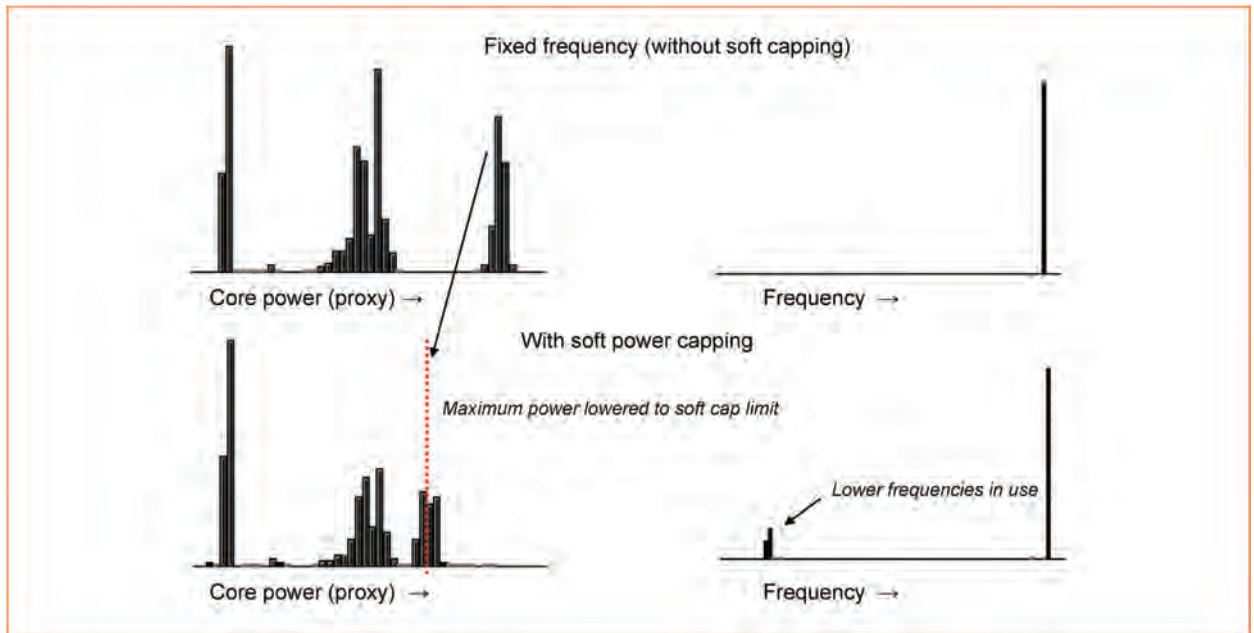


Figure 9

Soft capping: Histograms of power proxy and frequency for a variable workload.

per watt by shifting the allocated power between multiple processors or by carefully balancing the energy consumption between the processors and the memory. For example, if the turbo mode is selected, the chip will boost frequency up to 10% by leveraging the excess power capacity from either nonworst case workloads or excess capacity from unused cores that may be in an idle or reduced execution mode. Additionally, by using a power-proxy thresholding mechanism available in the POWER7 hardware, the EnergyScale microcontroller can implement a power-capping policy to enforce a hard ceiling on the system or even data-center power if desired while preserving optimal levels of performance on each processor core.

The ability to control the power consumption of individual cores is mandatory to be able to have a close control of the overall microprocessor chip power. *Power capping* policies limit the power of the system or of individual system components to a given maximum power-consumption budget. *Power shifting* [12] policies reallocate the power budget from one core to another or from processors to the memory in order to enhance the performance of critical tasks while still maintaining a global power budget. Fine-grain memory throttling and processor frequency scaling support in POWER7 enable power-constrained performance maximization technologies such as power shifting to be employed in the EnergyScale firmware in a production server environment for the first time.

One specific issue with the power capping and shifting policies is that the power consumed by a core at a given voltage and frequency may change very quickly as a function of its workload. A conservative approach to the power capping would assign each core a power budget based on worst case assumptions for a given operating point. Instead, given that the power proxy discussed earlier provides a per-core power-consumption estimate, the optimized solution to this problem offered by POWER7 is a fast control structure in the hardware that automatically controls the core frequency in response to workload changes in order to maintain the power-consumption budget. This control structure is depicted in **Figure 8**, along with the feedback loop and the power-proxy computation logic. The soft capping logic operates over discrete time intervals. In the current implementation, the power-proxy update and capping intervals range from 32 μ s to 524 ms in steps of 32 μ s. The frequency changes take place with a granularity of 28–420 MHz in 28-MHz steps. Note that the power cap implemented by this mechanism is “soft” in the sense that the response time and the estimation error render the power cap a statistical property rather than an absolute physical limit.

The control loop depicted in Figure 8 allows the firmware to specify the upper and lower thresholds for the power proxy, as well as the upper and lower step sizes for the frequency changes. Whenever the power proxy indicates a

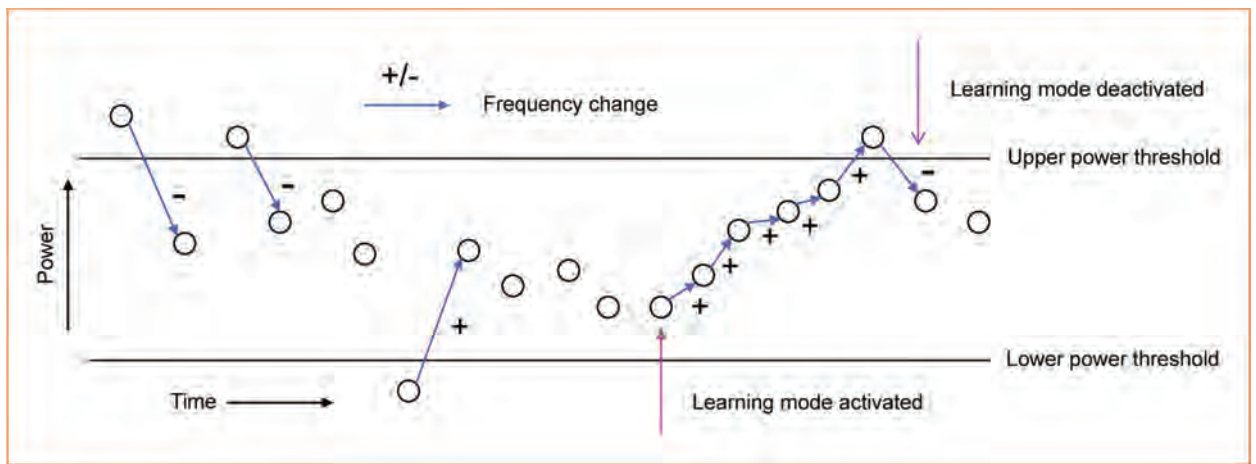


Figure 10

Power versus time with soft-power capping and learning mode.

power consumption that is above (below) the upper (lower) threshold, the controller reduces (increases) the core chiplet frequency within a certain frequency range. If the power consumption is within the upper and lower bounds, the frequency remains unchanged. Varying the threshold location, threshold separation, and frequency response allows several different soft capping schemes, including centering the estimated power consumption within a range or managing the statistical tail of the estimated power distribution above the upper threshold.

Figure 9 illustrates the soft capping mechanism in operation on a workload with the power varying per workload phase. Without soft capping, the frequency is constant at the nominal operating frequency. Once soft capping is enabled with a target below the maximum workload power, workload phases with higher estimated power trigger the automatic frequency reduction mechanism that tends to center the maximum power around the soft-power target.

As the separation between the upper and lower thresholds widens, the upper threshold tends to “harden” into a true power cap. However, workload changes may allow the current workload to operate at higher frequencies that were previously clipped by the soft capping logic. A *learning mode* allows the control circuit to reseek the maximum frequency that meets the given threshold settings, optimizing the performance within the power constraints for the current workload phase. **Figure 10** depicts a possible scenario highlighting the operation of the learning mode.

While the learning mode is enabled, the frequency increases during each interval even when the power-proxy value is within the power bounds. When the upper

threshold is exceeded, the learning mode is disabled, and the controller typically reduces the frequency again to end at a frequency that is close to the maximum frequency but still meets the estimated power constraints.

Instrumentation

POWER7 provides several on-chip hardware instrumentation enhancements to reduce the bandwidth and compute the power required by the external EnergyScale microcontroller, allowing it to have faster response times and, therefore, more efficient management of the energy.

Temperature calculation

Each POWER7 chip contains 44 digital thermal sensors (DTSs): five per each of the eight processor core chiplets and four others at the chip level in the SMP link and memory-controller regions. They are generally located near predicted hot spots on the chip for various workload types. An attempt is made to form a rough grid in the processor core chiplet such that we can build thermal maps for analysis. The center of the chip is not an area of high power density and can be extrapolated using all the core chiplets’ sensor 4 (sid4; see **Figure 11**).

In the DTS design, a temperature sensing circuit utilizes a band-gap diode voltage comparator similar to previous designs [13–15]. What is unique in POWER7 is that the voltage reading is converted by the logic on the chip to a temperature in degree Celsius via a polynomial curve fit $px^2 + mx + b$, where x is the output of the comparator after the analog-to-digital conversion. The coefficients are derived during the manufacturing test and calibrated per DTS using a traditional off-chip thermal diode as a reference. The calculation in the hardware provides an optimal

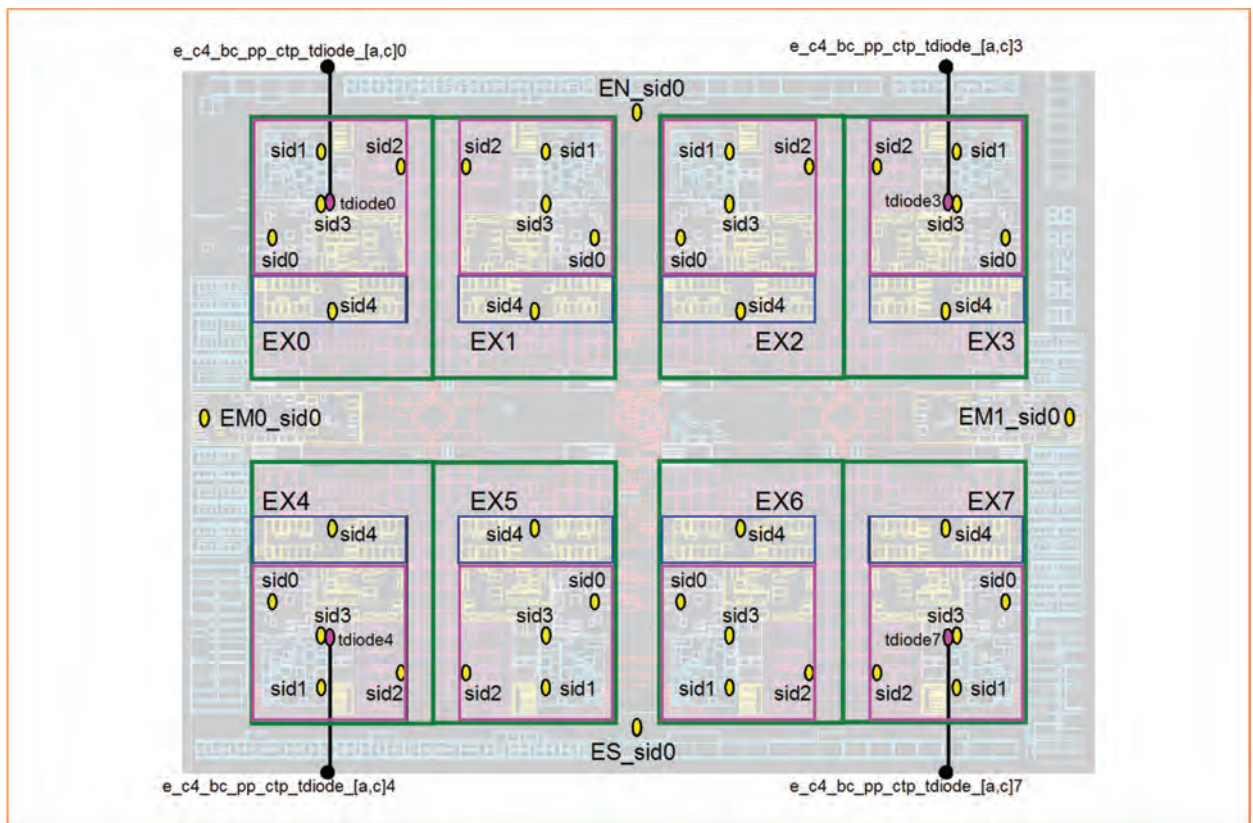


Figure 11

DTS marked on the POWER7 chip floor plan, along with the four thermal diodes used for calibration. (Green) Core chiplets (named EX0 to EX7) containing (magenta) the processor cores and (blue) the L2 caches.

quadratic curve fit across the required chip operating range in the system, centered on the expected 65 °C to 80 °C operating temperature of the chip. The firmware can then use the temperature values from the hardware directly without expending a valuable real-time computing resource. The temperature measurements are used to generate the power proxy and also to prevent the chip from possible overheating during turbo operation.

OCA monitoring infrastructure

Power-management firmware access to sensor information and control of actuators on the processor is done via an on-chip serial communication infrastructure called *SCOM*. The READ and WRITE operations to select SCOM register addresses provide the sensor data and allow the software to manipulate the available control mechanisms during the runtime. An I2C interface provides connectivity from the power-management microcontroller to each POWER7 chip. The I2C is a serial protocol with some overhead, and the bus is limited to 400-Kb/s transfer rates. With the growth in the sensor traffic from the POWER6 generation—both

in the kinds of items tracked and in their number (e.g., four times the number of cores)—a lower overhead access method was required. POWER7 implements an OCA unit to provide this function.

The OCA helps the power-management firmware in two ways.

- A collection of SCOM registers can be sent as a batch transaction to the external microcontroller with streaming support for a table of prespecified register addresses. This significantly reduces both the bandwidth (amortizing the SCOM address transmission over the collection) and processing overheads for the sensor data collection (processing them as a batch instead of individually), improving the scalability of the firmware.
- A collection of sensor values from the SCOM registers to be streamed out is handled by the OCA with a firmware control on inter-event times and periodicity of the collection. This improves the temporal correlation of information from the different sensors.

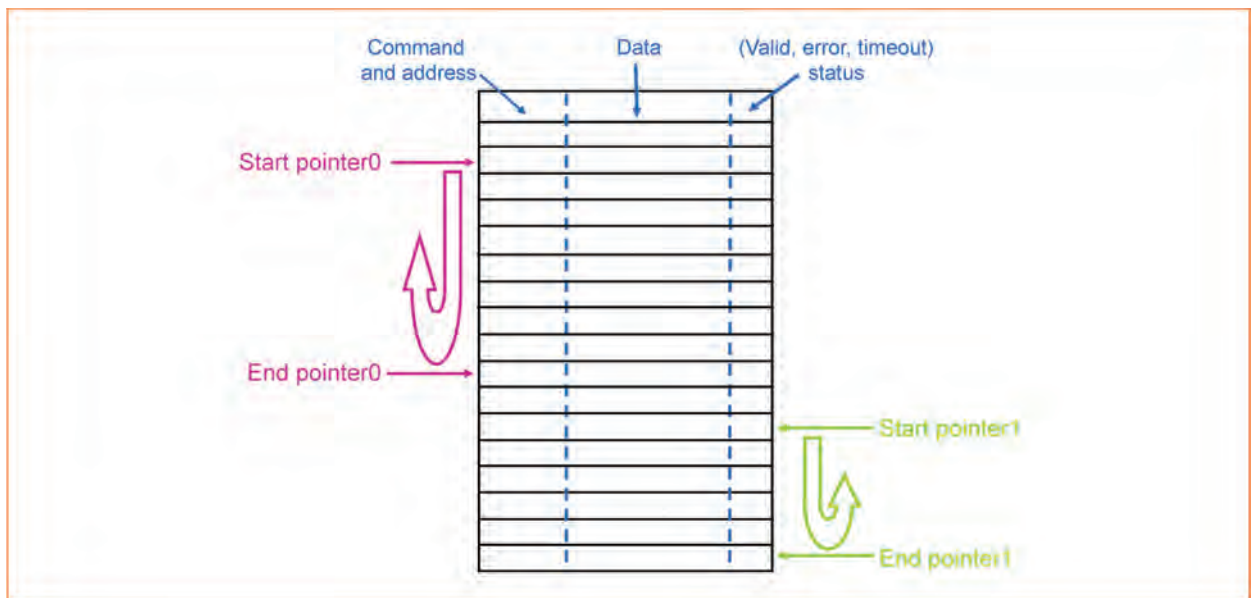


Figure 12
OCA structure.

The OCA collects power, performance, and temperature information from throughout the POWER7 chip using the same on-chip communications infrastructure utilized by other devices in the system. The OCA stores the information into a central location in response to a precise programmable schedule where it is periodically streamed out to the off-chip microprocessor on demand. While the OCA is primarily intended to facilitate the firmware access to on-chip sensors for the system power–performance management, it is also designed to perform more complex off-chip sensor access sequences that include WRITE operations for register address reprogramming, should an external I2C-based measurement device be present in the system.

As shown in **Figure 12**, the OCA implementation consists of a table and two identical sequencers. The control logic supports the aforementioned streaming READ mode of the table contents. A table entry consists of an encoded command, a compressed SCOM register address, and a data field to be used by the READ or WRITE command. As it processes each entry, the OCA stores the result and possible-error conditions in the status area of that entry. When streaming out the table entries, only the data is streamed out. This additionally minimizes the bandwidth requirements since the command and address information is known to the external entity performing the READ command. Finally, the error codes of the streamed entries are dynamically merged into one additional data packet that the OCA provides at the end of the streaming READ command.

Each sequencer is set up to process lines from the 128-entry table. The user can configure how many and which entries are processed by each sequencer, along with delay settings, which are separately configurable over several time scales. An additional feature of the sequencers is the possibility to poll any SCOM addressable register for an expected value. The polling command can be written to the table just like any other READ or WRITE command. This allows more complex operations to be performed when setting up and reading the sensor data.

Work-rate stall measurement

With modern microprocessors, the impact of the frequency on the application performance is highly dependent on the specific workload characteristics, particularly its memory hierarchy usage. For memory-bound workloads, a relatively large decrease in the frequency could only have a small impact in the performance leading to significant gains in energy efficiency at the lower frequencies, particularly when coupled with the correspondingly lower voltage. Prior research [16] has shown the benefits of using a processor’s stall rate measurements for runtime power–performance tradeoffs using user-programmable performance counters, but POWER7 is the first server processor with dedicated stall counter support for the energy management.

The POWER7 work-rate stall counter is designed to track the extent of stall cycles associated with the load–store

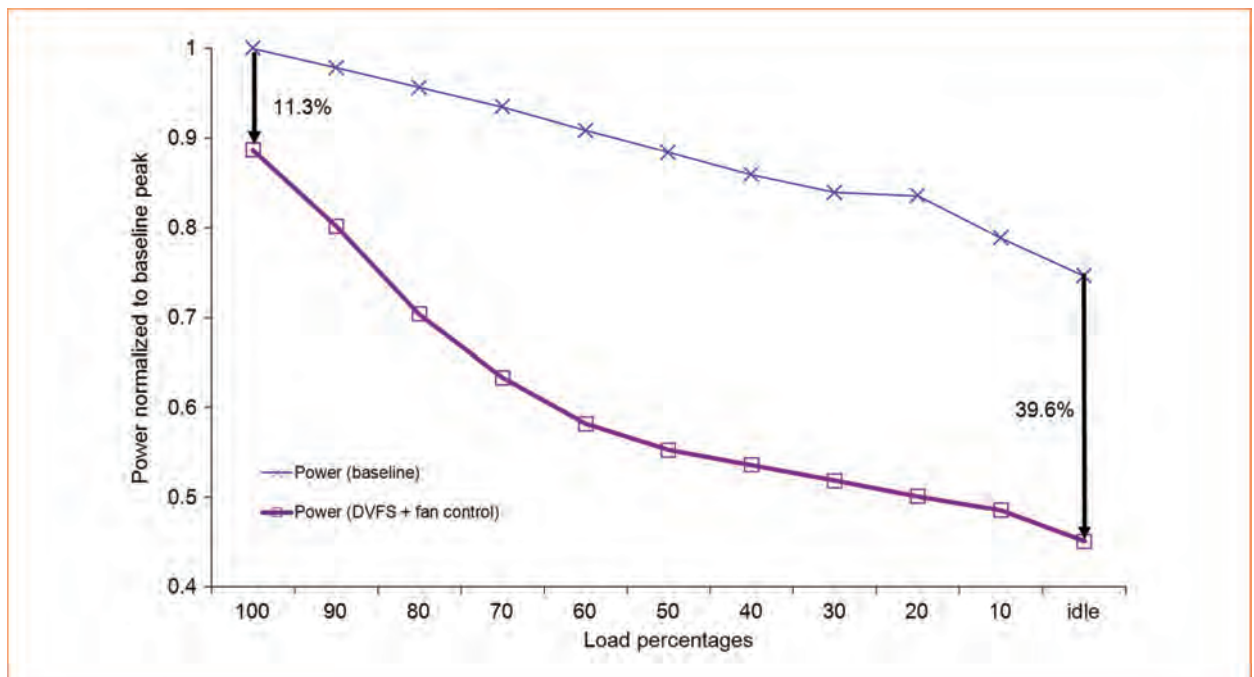


Figure 13

System-level power reduction on a prototype four-node POWER7 system using the DVFS and the energy-aware thermal management and dynamic fan control.

unit misses that prevent subsequent instructions from completing. The firmware can periodically sample the work-rate stall counter for each core every decision interval (e.g., every 10 ms) to determine the relative importance of the frequency to the core's instruction execution rate. The programmable memory hierarchy event counts, together with a completed instruction counter in POWER7, can further help to fine tune the estimates of the frequency impact on the instruction rate. The completed instruction counter and the programmable memory hierarchy event counters also exist in POWER6. The increased out-of-order design and multiple outstanding memory miss handling capabilities of POWER7 demanded the addition of the work-rate stall counter for better runtime performance impact estimation.

Power-efficiency algorithms and results

A key distinction with most of the other server platforms, where some forms of similar energy-management approaches such as dynamic voltage-and-frequency scaling (DVFS) have been used, as described below, is the support for and exploitation of both monitoring and control logic from the firmware. This approach renders the Power Architecture-based EnergyScale solutions relatively free of operating system idiosyncrasies for enabling power

management and also more easily applicable to multi-partition systems in virtualized environments. First introduced in POWER6 systems, the EnergyScale dynamic power management takes a step forward with the added features of POWER7. Per-core frequency scaling, faster voltage scaling support, extended voltage–frequency slewing range, finer grain memory access controls, extended memory low-power mode support, and programmable logic for sensor acquisition all make POWER7 a powerful vehicle for dynamic adaptive power management. Power-management functions are delivered through a combination of monitoring and control algorithms implemented in the service processor firmware—in both the node-level power-management microcontrollers and the general service processors—and energy-aware processor resource management by the Hypervisor and operating systems.

There are two kinds of power-management functions: those that maintain the system operation within the power or thermal limits and those that detect slack in the system usage either to lower the energy consumption by utilizing lower power active and idle states or to boost performance by taking advantage of the guard-band built into the system operational parameters when not running under worst case conditions. As in POWER6, the EnergyScale microcontroller monitors several on-chip counters to measure the utilization

of the system and varies both the frequency and the voltage of the chip accordingly during nonidle periods of activity. This intelligent DVFS allows POWER7 to maintain the system throughput while reducing the energy consumption of the processor chip significantly. Initially, the power is reduced at an almost cubic rate due to the ability to decrease both the voltage and the frequency to the core chiplets. Once $V_{\min}$, i.e., the minimum guard-banded voltage at which the chip will reliably function, is reached, the EnergyScale microcontroller continues to reduce the frequency to provide the linear power take down of the core chiplet power.

The functions that manage to reach within thermal limits are more energy efficient themselves by regulating the fan speeds to consume just enough fan power to maintain desired thermal characteristics and no more. Algorithms for the power capping functions are engineered to maximize the performance within a specified power envelope by combining real-time power measurements and subsystem activity measures as the feedback, along with the DVFS and the memory throttling as actuators implementing a power shifting logic for the performance-conserving capping. The functions for energy saving are also made performance aware by incorporating appropriate slack detection methods. Optionally, they can also be used to boost the performance by operating at higher than nominal frequencies, given that the power capping controls are in place for dynamic protection when needed. **Figure 13** shows the impact of having both an energy-aware thermal control loop and a slack-aware DVFS-based energy saving loop act in conjunction. The baseline shows the variation in the system power with load for a specific workload on a prototype four-socket POWER7 system. The “DVFS + fan control” shows the power or the energy savings obtained by the combination with no impact to throughput at all load levels. Details on the algorithms used for DVFS management and fan control have been previously published [17].

Processor resource management by the Hypervisor and operating systems consist mainly of utilizing the nap- and sleep-mode variants for idling cores. Operating systems can use different strategies for using these modes. Using nap for short idles and sleep when average load levels indicate that cores could be freed up for longer periods of time (such as when a dedicated partition folds one of its cores) are the initial planned use on the IBM POWER* platform. An idle-time history-based selection of appropriate low-power mode could also be used, using latency specifications associated with the modes. Each approach can be further made configurable through operating system- and firmware-level tunable parameters allowing the customer to customize them for each workload.

Conclusion

The POWER7 processor’s power-management capabilities build on experience gained with the previous-generation

POWER6 implementation and incorporate the latest research results on the active energy management for large systems. POWER7 extends the firmware ability to finely tune the energy-and-performance tradeoffs through transparent per-core frequency scaling; autonomous frequency control; and enhanced power, performance, and thermal sensor implementations. Mechanisms such as the OCA, the temperature conversion to degree Celsius, and the power-proxy calculations built into the hardware assist the hard real-time power-management firmware by offloading repetitive calculations. The introduction of a second lower power idle mode has minimized the system power during periods of inactivity while providing low-latency capacity on demand in virtualized environments. Multiple voltage control interfaces have met the distinct needs of lower end and high-end systems. The design of POWER7 has also anticipated the power-management firmware and software evolutions by supporting privileged communications between the EnergyScale microcontroller and the POWER Hypervisor for cooperative management. The combination of the EnergyScale firmware with advanced features of the POWER7 hardware allows IBM POWER systems to offer world-class energy-management capability for a smarter planet.

Acknowledgments

Some material in this paper is based on the work supported by the Defense Advanced Research Projects Agency under its Agreement HR0011-07-9-0002. Statements regarding chip and system features do not imply that IBM will introduce a system with this capability.

*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 AMD Corporation in the United States, other countries, or both.

References

1. H.-Y. McCreary, M. A. Broyles, M. S. Floyd, A. J. Geissler, S. P. Hartman, F. L. Rawson, III, T. J. Rosedahl, J. C. Rubio, and M. S. Ware, “EnergyScale for IBM POWER6 microprocessor-based systems,” *IBM J. Res. & Dev.*, vol. 51, no. 6, pp. 775–786, Nov. 2007.
2. M. S. Floyd, S. Ghiasi, T. W. Keller, K. Rajamani, F. L. Rawson, III, J. C. Rubio, and M. S. Ware, “System power management support in the IBM POWER6 microprocessor,” *IBM J. Res. & Dev.*, vol. 51, no. 6, pp. 733–746, Nov. 2007.
3. J. Dorsey, S. Searles, M. Ciraula, S. Johnson, N. Bujanos, D. Wu, M. Braganza, S. Meyers, E. Fang, and R. Kumar, “An integrated quad-core Opteron processor,” in *Proc. IEEE ISSCC Dig. Tech. Papers*, 2007, pp. 102–103.
4. J. A. Tierno, A. V. Rylyakov, and D. J. Friedman, “A wide power supply range, wide tuning range, all static CMOS all digital PLL in 65-nm SOI,” *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 42–51, Jan. 2008.
5. IBM Corporation, *Power ISA Version 2.05*. [Online]. Available: http://www.power.org/resources/reading/PowerISA_V2.05.pdf

6. C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," in *Proc. Int. Symp. Microarchitecture (MICRO)*, 2003, p. 93.
7. B. Stackhouse, S. Bhimji, C. Bostak, D. Bradley, B. Cherkauer, J. Desai, E. Francom, M. Gowan, P. Gronowski, D. Krueger, C. Morganti, and S. Troyer, "A 65 nm 2-billion transistor quad-core Itanium processor," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 18–31, Jan. 2009.
8. R. Jotwani, S. Sundaram, S. Kosonocky, A. Schaefer, V. Andrade, G. Constant, A. Novak, and S. Naffziger, "An x86-64 core implemented in 32 nm SOI CMOS," in *Proc. IEEE ISSCC Dig. Tech. Papers*, 2010, pp. 106–107.
9. K. Choi, R. Soma, and M. Pedram, "Dynamic voltage and frequency scaling based on workload decomposition," in *Proc. ISLPED*, 2004, pp. 174–179, DOI: 10.1145/1013235.1013282. [Online]. Available: <http://doi.acm.org/10.1145/1013235.1013282>
10. G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott, "Energy efficient processor design using multiple clock domains with dynamic voltage and frequency scaling," in *Proc. 8th Int. Symp. High-Perform. Comput. Archit.*, 2002, pp. 29–40.
11. N. AbouGhazaleh, A. Ferreira, C. Rusu, R. Xu, B. R. Childers, R. Melhem, and D. Mosse, "Integrated CPU and L2 cache frequency/voltage scaling using supervised learning," in *Proc. HiPEAC Workshop Stat. Mach. Learn. Approaches Appl. Archit. Compilation (SMART)*, SIGPLAN Notices, 2007, vol. 42, no. 7, pp. 41–50, DOI: 10.1145/1273444.1254773. [Online]. Available: <http://doi.acm.org/10.1145/1273444.1254773>
12. W. M. Felter, K. Rajamani, T. W. Keller, and C. Rusu, "A performance-conserving approach for reducing peak power consumption in server systems," in *Proc. 19th ACM ICS*, 2005, pp. 293–302, DOI: 10.1145/1088149.1088188. [Online]. Available: <http://doi.acm.org/10.1145/1088149.1088188>
13. M. Yoshida and D. W. Boerstler, "Thermal sensing circuit using bandgap voltage reference generators without trimming circuitry," U.S. Patent 7 789 558, Sep. 7, 2010.
14. T. H. Lee, M. G. Johnson, and M. P. Crowley, "Temperature sensor integral with microprocessor and methods of using same," U.S. Patent 5 961 215, Oct. 5, 1999.
15. J. G. O'Dwyer, "Apparatus and method for temperature measurement using a bandgap voltage reference," U.S. Patent 7 225 099, May 29, 2007.
16. K. Rajamani, H. Hanson, J. Rubio, S. Ghiasi, and F. Rawson, "Application-aware power management," in *Proc. IEEE IISWC*, 2006, pp. 39–48.
17. K. Rajamani, F. Rawson, M. Ware, H. Hanson, J. Carter, T. Rosedahl, A. Geissler, G. Silva, and H. Hua, "Power-performance management on an IBM POWER7 server," in *Proc. 16th ACM/IEEE ISLPED*, 2010, pp. 201–206, DOI: 10.1145/1840845.1840885. [Online]. Available: <http://doi.acm.org/10.1145/1840845.1840885>

Received March 3, 2010; accepted for publication January 5, 2011

Michael Floyd IBM Systems and Technology Group, Austin, TX 78758 USA (mfloyd@us.ibm.com). Mr. Floyd received a bachelor's degree in computer engineering with highest honors from the Georgia Institute of Technology in 1995 and a master's degree from Stanford University in 2000. He is the Architect and Lead for the POWER7 EnergyScale design. His experience with IBM server development includes hardware bring-up, test, debug, and RAS in addition to design, lead, and microarchitect roles on the POWER4*, POWER5*, and POWER6 processors and support chips. He is an IBM Master Inventor with more than 40 issued U.S. patents and as many still pending.

Malcolm Ware IBM Research Division, Austin Research Laboratory, Austin, TX 78758 USA (mware@us.ibm.com). Mr. Ware received a bachelor's degree in electrical engineering from Purdue University in 1983 and joined IBM the same year. He received his

master's degree from North Carolina State University in 1986. He is a member of the Power-Aware Systems Research Group. He has been researching dynamic power management for the last 7 years for all classes of IBM servers, including Intel, POWER6, POWER7, and z7* processors. Prior to working on power management, he spent 20 years on digital signal processor technology and was on assignment at IBM's Zurich Research Laboratory for 7 of those years. He is an IBM Master Inventor.

Karthick Rajamani IBM Research Division, Austin Research Laboratory, Austin, TX 78758 USA (karthick@us.ibm.com).

Dr. Rajamani graduated with a B.Tech. degree in electronics and communications from IIT Madras and obtained his master's and Ph.D. degrees in electrical and computer engineering from Rice University, Houston. He works in the Power-Aware Systems Research Group, where he has made key contributions to different dynamic power management architectures from embedded systems to high-end servers including designs of the IBM POWER6 and POWER7 processors and systems and the IBM EnergyScale technology. His current research includes the design of energy-efficient, high-performance memory systems, energy-aware systems software, middleware, and management tools and energy-efficient datacenters.

Tilman Gloekler IBM Research Division, 71032 Boeblingen, Germany (gloekler@de.ibm.com). Dr. Gloekler received his diploma degree in electrical engineering with honors from the University of Stuttgart, Germany, in 1997, and his Ph.D. degree from Aachen University of Technology, Germany, with honors in 2003. He is the design lead for the POWER7 chip power management implementation.

His professional experience includes design, verification, and bring-up both for signal processing and high-performance microprocessor chips. Before he joined the Power Management team, he was the lead for the pervasive logic verification of the IBM Cell processor.

Bishop Brock IBM Systems and Technology Group, Austin, TX 78758 USA (bcbrock@us.ibm.com). Mr. Brock holds B.S., B.A., and M.S. degrees from the University of Texas at Austin. He is currently a Senior Engineer in the IBM Systems and Technology Group in Austin, Texas, having previously served as a Research Staff Member at IBM Research, Austin. His contributions to the POWER7 project include the development of hardware verification, microarchitectural validation, and power/performance prediction methodologies for power management hardware and firmware systems.

Pradip Bose IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (pbose@us.ibm.com).

Dr. Bose holds M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana Champaign. He is a Research Staff Member and Manager of the Reliability and Power-Aware Microarchitecture department at the IBM T. J. Watson Research Center. Dr. Bose has been involved in pre-silicon modeling and definition of IBM POWER series microarchitectures, beginning with the research precursor of the very first RS/6000* product. He is an IBM Master Inventor and the recipient of numerous IBM internal awards. He is an IEEE Fellow.

Alper Buyuktosunoglu IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (alperb@us.ibm.com). Dr. Buyuktosunoglu received his Ph.D. degree in electrical and computer engineering from the University of Rochester. Currently, he is a Research Staff Member in the Reliability and Power-Aware Microarchitecture department at IBM T. J. Watson Research Center. He has supported the design of IBM pSeries* and zSeries* microprocessors in the areas of power-aware computer architectures, dynamic power management, and high-level power/performance modeling. He has more than 25 issued or pending patents, has received several IBM-internal awards, has published more than

30 papers, and has served on various conference technical program committees in these areas. He is a senior member of the IEEE and is currently serving on the editorial board of *IEEE Micro*.

Juan C. Rubio *IBM Research Division, Austin Research Laboratory, Austin, TX 78758 USA (rubioj@us.ibm.com)*. Dr. Rubio received a B.S. degree in electrical engineering from Universidad Santa Maria La Antigua, Panama, in 1997, and an M.S. and Ph.D. degree in computer engineering from the University of Texas at Austin in 2004. He is a Research Staff Member at IBM Research, Austin. He is interested in computer system architecture, control systems and performance analysis. At IBM, he explores techniques to model, monitor and manage power, temperature, and performance in computer servers and data centers. He worked on power management functions for IBM's POWER6, POWER7, and xSeries* systems and was part of the team that designed and developed the IBM EnergyScale technology.

Birgit Schubert *IBM Systems and Technology Group, 71032 Boeblingen, Germany (bschubert@de.ibm.com)*. Mrs. Schubert joined IBM in 1979 and has spent most of her time working in the area of processor development. Her responsibilities have included logic design for various design units for zSeries and pSeries chips, as well as bring-up and chip integration tasks. She was on assignment in Austin, Texas, from 2006 to 2009 where she joined the Pervasive and Power Management Team.

Bruno Spruth *IBM Systems and Technology Group, 71032 Boeblingen, Germany (spruthb@de.ibm.com)*. Mr. Spruth received a Dipl.-Ing. degree in electrical engineering from the University of Stuttgart in 2006. He is a member of the Pervasive and Power Management Team in the IBM Systems and Technology Group. Since joining IBM, he has been working on hardware design and timing for both pSeries and zSeries processor chips and served as the pervasive function bring-up lead on POWER7.

Jose A. Tierno *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (tierno@us.ibm.com)*. Dr. Tierno holds M.S. and Ph.D. degrees from the California Institute of Technology. He is a Research Staff Member and Manager of the mixed signal design group. His interests are in the design of digital replacements for analog circuits, in particular all-digital phase-locked loops, and the design of self-timed digital circuits. He has received several Invention Plateau and internal awards from IBM.

Lorena Pesantéz *IBM Systems and Technology Group, Austin, TX 78758 USA (mlpesant@us.ibm.com)*. Ms. Pesantéz received a bachelor's degree in electrical engineering from the University of Texas at Austin in 2008. She is a member of the Server Hardware Performance team, where she specializes in POWER instrumentation. She aided in the definition, hardware data collection, and tuning of the power proxy for the POWER7 chip.

Design methodology for the IBM POWER7 microprocessor

The IBM POWER7[®] microprocessor, which is the next-generation IBM POWER[®] processor, leverages IBM's 45-nm silicon-on-insulator (SOI) process with embedded dynamic random access memory to achieve industry-leading performance. To deliver this complex 567-mm² die, the IBM design team made significant innovations in chip design methodology. This paper describes the most critical methodology innovations specific to POWER7 design, which were in modularity, timing closure, and design efficiency.

J. Friedrich
R. Puri
U. Brandt
M. Buehler
J. DiLullo
J. Hopkins
M. Hossain
M. Kazda
J. Keinert
Z. M. Kurzum
D. Lamb
A. Lee
F. Musante
J. Noack
P. J. Osler
S. Posluszny
H. Qian
S. Ramji
V. Rao
L. N. Reddy
H. Ren
T. Rosser
B. R. Russell
C. Sze
G. Téllez

Introduction

The next processor of the POWER* family, which is called POWER7*, contains eight quad-threaded cores integrated with two memory controllers and high-speed system links on a 567-mm² die, employing 1.2 billion transistors in 45-nm complementary metal-oxide semiconductor (CMOS) silicon-on-insulator (SOI) technology [1] and embedded dynamic random access memory (eDRAM) technology. The POWER7 design team leveraged significant innovations in design methodology, such as modularity, timing closure, and design efficiency, that were critical in delivering industry-leading performance on a very aggressive schedule. This paper describes these critical methodology innovations that were specifically developed for POWER7 and future designs.

Modularity

Because POWER7 is a 567-mm² die containing eight 4-way simultaneous multithreaded cores, a 32-MB eDRAM Level 3

(L3) cache, two memory controllers, and high-bandwidth symmetric multiprocessing (SMP) interfaces, it represented a significant integration challenge. To enable parallel design closure, we partitioned the chip into independent modular blocks that contained multiple units, a complete power and clocking grid, and the logic to initialize and test the block. This modular approach enabled independent manufacturing testing of the blocks and significant recovery of partial good die with errors in cores or cache subregions. Modularity also enables the reuse of blocks on other designs without reintegration. **Figure 1** illustrates the location of the four largest components of POWER7: EX [core, Level 2 (L2) cache, and one local 4-MB L3 cache], EN (local SMP), ES [(remote SMP and GX input/output (I/O)], and EM (memory controller).

Modular blocks also allowed POWER7 to provide new levels of system flexibility by maintaining 23 frequency domains, 47 input voltages, and 16 generated voltages. Each of the eight EX blocks is powered by separate voltages for logic, static random access memory (SRAM), and DRAM, as well as an analog voltage for the digital phase-locked loop. Each EX unit also contains two charge-pump-derived

Digital Object Identifier: 10.1147/JRD.2011.2105692

©Copyright 2011 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied by any means or distributed royalty free without further permission by computer-based and other information-service systems. Permission to republish any other portion of this paper must be obtained from the Editor.

0018-8646/11/\$5.00 © 2011 IBM

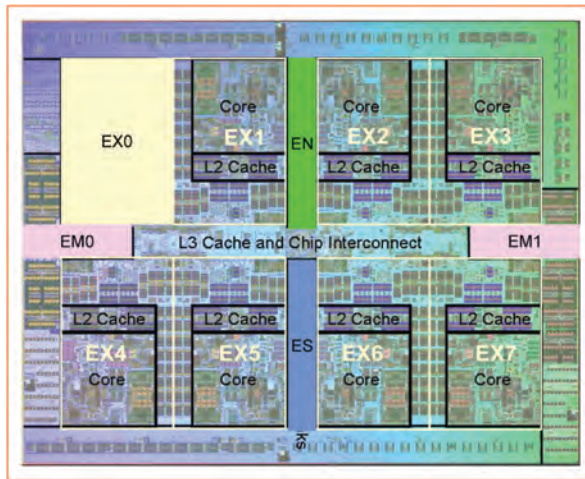


Figure 1

Major POWER7 modular blocks.

voltages to supply the DRAM word lines. The 11 distinct I/O and nest voltage regions enable packages not using the full off-chip bandwidth to save approximately 20% of nest power by grounding the supplies for one memory interface and three of the SMP buses. **Table 1** shows the voltage domains and operating ranges of POWER7.

Each core can independently operate, enabling power/performance optimization by setting individual core frequencies at nominal, power save, turbo, or sleep, an architected state that disables clocks and enables retention voltage. The 32-MB shared L3 is divided into eight high-speed local regions that each operate at half the speed of the associated core. These L3 regions, namely, the 6.4-Gb/s memory interface, the 3-Gb/s SMP interfaces, and the 3-Gb/s GX I/O, communicate asynchronously across a fixed-frequency on-chip interconnect domain. **Figure 2** shows the frequency domains of POWER7.

Timing

Another key area of methodology innovation in POWER7 was timing closure. Significant enhancements in both macro and global timing enabled first-pass silicon that achieved a robust voltage operating window and approached final product frequencies. Additionally, the innovations allowed static timing rules for all on-chip arrays for the first time, eliminating the need for timing rule generation using time-consuming and error-prone manual cross sections and greatly improved silicon quality.

Macro timing

The IBM transistor-level static timer, i.e., EinsTLT, reads in the netlist (schematic or extracted), flattens it down to the transistor level, and partitions it into channel-connected

components (CCCs) [2]. Each CCC is analyzed using a novel state-analysis method in order to construct the timing model (propagation arcs and timing checks) that is used within a static timing engine (EinsTimer*). EinsTLT also uses an embedded circuit simulator (Adaptively Controlled Explicit Simulation [2]) to propagate waveforms through CCCs.

Certain topologies encountered in POWER7 arrays required special handling such as the following:

- 1) read–write bit columns modeled using multiple-timing modes (MTMs);
- 2) redundancy multiplexers (MUXs) (both data in and data out; some modeled as static circuits and others as dynamic circuits);
- 3) level-shifter circuits with feedback to turn precharge high prematurely to avoid contention during evaluate;
- 4) reset-before-evaluate domino;
- 5) pulsed-clock circuits; and
- 6) deliberately delayed clocks feeding domino or gating circuits requiring a different set of cycle adjusts than normal clocks.

In addition, a new feature in the tool, which is called *differential timing*, was used in properly timing differential circuits in I/O macros.

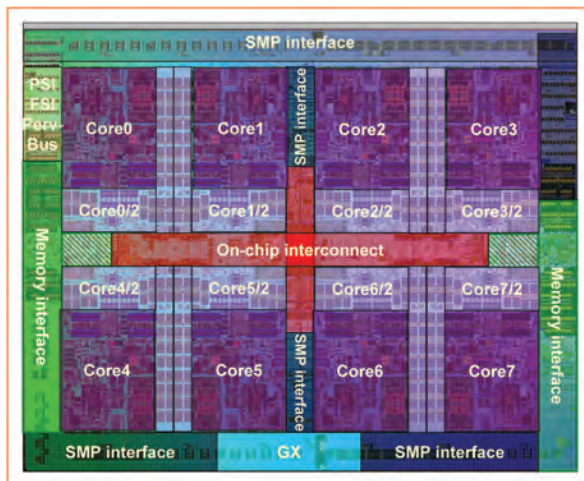
Used for the first time in the POWER7 methodology, the MTM method is a novel technique developed to model complex circuits in different modes, where, in each mode, the model is reduced to a simpler case that the EinsTLT analysis code can handle well. For example, consider the read–write bit columns. EinsTLT creates two separate modes. In “read” mode, the “write” circuitry is disabled, and the local bit line “reading” a 0 from the cell whose word line is turned on is modeled as a simple domino NOR. In “write” mode, the cells are temporarily turned off, and the structure is modeled as a multioutput domino circuit that propagates timing to the local bit lines. From here, a separate step propagates timing to the cell storage nodes, and special checks are created to ensure correctness in timing.

The level-shifter circuits in POWER7, as shown in **Figure 3(a)**, were timed using a two-pass method due to a timing loop: $w3 \rightarrow w0 \rightarrow \text{clkout} \rightarrow \text{clkoutb} \rightarrow \text{fbk} \rightarrow w3$. The timing diagram is shown in **Figure 3(b)**.

The procedure used in EinsTLT to model the level shifter is as follows: First, a timing template is used to create a special model for the NAND2 gate. In this special model, the feedback loop is broken by not allowing fbk to propagate directly to the output w3, but rather to an internal timing point named w3_int. The simulator probe for w3_int is the same as that for the output w3; thus, the timing on w3 would be equivalent to that on w3, except that the loop is broken. In addition, in the special model, the w2 input of the NAND2 gate propagates both edges rise $\rightarrow$ fall and fall $\rightarrow$ rise to the w3 output, and for both these propagations, the sensitizations force $\text{fbk} = 1$. First-pass timing is then performed; note that the feedback loop is broken. At the end of this first-pass timing, w3 will rise according to the dotted-line waveform shown in **Figure 3(b)**. However, the w3_int node will rise, as shown by the gray waveform propagating from fbk fall, which eventually is triggered by

Table 1 POWER7 chip voltage domains. (DPLL: digital phase-locked loop.)

<i>Name</i> (# = 0 to 7)	<i>Meaning</i>	<i>Type of voltage</i>	<i>Minimum circuit voltage</i>	<i>Maximum circuit voltage</i>
$V_{dd_EX\#}$	Logic supply for chiplet #	Adaptive, dynamic	0.7	1.3
$V_{CS_EX\#}$	Array supply for chiplet #	Adaptive, dynamic	0.8	1.3
$V_{DR_EX\#}$	DRAM supply for chiplet #	Adaptive, dynamic	0.8	1.3
$V_{pp_EX\#}$	Charge pump-derived positive DRAM wordline rail	Derived	1.6	2.0
$V_{WL_EX\#}$	Charge pump-derived negative DRAM wordline rail	Derived	-0.4	0
$VDPLL_EX\#$	# DPLL supply for chiplet #	Fixed, static	1.62	2.0
V_{dd_NEST}	Logic supply for nest and pervasive logic	Fixed, static	1.0	1.2
$V_{dd_MC0/1}$	Logic supply for MC	Fixed, static	1.0	1.2
V_{SB}	Supply for pervasive	Fixed, static	1.1	1.3
$V_{mem0/1}$	Supply for differential I/O	Fixed, static	0.95	1.15
V_{io_w}	Supply for EI3 I/O w-bus	Fixed, static	1.0	1.2
V_{io_xyz}	Supply for EI3 I/O xyz-bus	Fixed, static	1.0	1.2
V_{io_abrx}	Supply for EI3 I/O A&B receiver	Fixed, static	1.0	1.2
V_{io_abtx}	Supply for EI3 I/O A&B transmit	Fixed, static	1.0	1.2
V_{io_c}	Supply for EI3 I/O C bus	Fixed, static	1.0	1.2
$AV_{dd_PB},$ $AV_{dd_ABC},$ $AV_{dd_M0},$ AV_{dd_M1}	PLL analog supplies	Fixed, static	1.62	2.0

**Figure 2**

POWER7 frequency domains.

clk_{in} fall [follow the blue propagations in Figure 3(b)]. Since this w₃_int rise time is in the next cycle, a cycle adjust is needed to move it back to the current cycle [as shown by the red arrow in Figure 3(b)], which results in the following: The above adjustment is applied to the rise of w₃ after the first-pass timing, and then, the second-pass timing is performed. During this second-pass timing, the waveform on w₃ will be according to the solid lines shown in Figure 3(b), and this will result in no contention with clk_{in} on the domino circuit, which, in turn, will produce the correct timing at the clk_{out} output that propagates to the core array. A variety of different level-shifter topologies have been handled using this two-pass timing technique.

POWER7 also had two register-file macros, namely, vector register file and general-purpose register, which used “double-pumped” logic in the arrays. This poses problems for static timing tools, in general, because the memory cell appears to be triggered from both the leading and the trailing edge of the ideal clock pulse. Some novel techniques using MTM were used again to separate the influence of

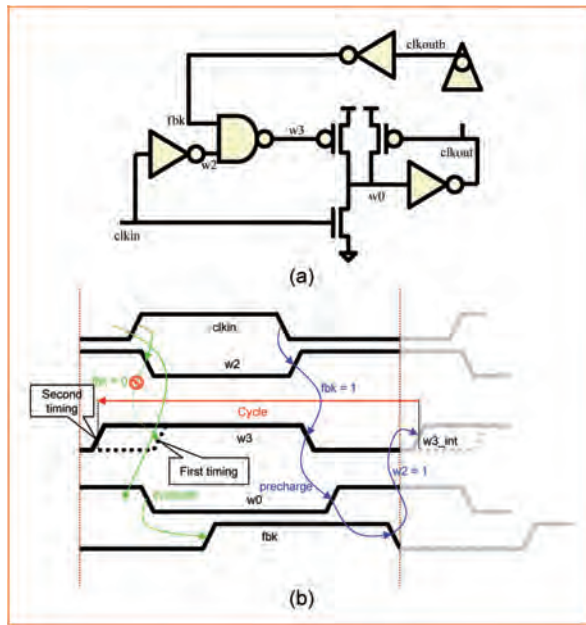


Figure 3
(a) Level-shifter circuit (invert type) used in the POWER7 array. (b) Timing diagram for the level shifter.

the two clocks used in the double pumping. A check was introduced to ensure that the two clock pulses do not overlap; a small predefined overlap was allowed. Prior to POWER7, such designs were timed using a manual and error-prone “gray-box” methodology that sometimes led to a failure in the test hardware. Therefore, in POWER7, a huge effort was put into getting as many arrays as possible timed through native EincTLT without the use of these gray boxes.

Global timing methodology

In POWER7, higher chip density, additional logic hierarchies, increasing asynchronous design interfaces, new requirements for power, and other timing goals brought new timing challenges and advancements in response to those challenges. We discuss these advancements and how they shaped IBM common global timing tools and methodology for microprocessors. A more general description of the POWER timing methodology and timing closure methods and phases can be found in [3, 4].

The addition of eight cache *chipslets* in POWER7 presented the chip-timing model with a new level of design hierarchy and increased model size. To achieve rapid chip-timing turnaround early in the design process, it was necessary to selectively create a model containing a single processor core along with all representations of the cache chipslets and present it to the chip-timing run for analysis. The implementation was a model selection, unique instance

identification, rename, and deletion process of the logic not to be timed. This scheme enabled the chip-timing process to progress at a fast rate during the estimated timing phase.

The POWER7 chip expanded the use of asynchronous interfaces, prompting special handling of these crossings in timing analysis, where a functional or performance requirement existed. Methodologies to validate the latency and the skew of the asynchronous boundary crossings in the chip were developed. The latency check was used to validate the arrival time of the asynchronous signal at the receiving latch point relative to the sourcing clock. The skew was measured by calculating the largest arrival time difference at different destination latches within any single clock domain, and it was checked that it was no greater than one clock cycle of the receiving clock. Because asynchronous crossing signals are phase-pair excluded in timing analysis due to the fact that no synchronous timing relationship exists, a new report was implemented to detect where these phase-pair exclusions occurred at latch points in the design and to gather arrival time and phase information at those points. The latency and skew checks were then performed by side processes outside the timing run to ensure all asynchronous boundaries met the design constraints.

The use of a database to store timing reports and information became more prevalent in POWER7. The IBM View Late Timing Report (VLTR) again enabled storage and retrieval of timing report information using a DB2* database. VLTR was used in the chip-timing closure process to assign path ownership, to document timing path solutions, and to generate timing statistics and histograms to show timing takedown progress throughout the design. Additional extensions were made to the database to store power information (CADT). The timing methodology incorporated on POWER7 a method to extract integration buffer types and counts per clock domain, as well as wire capacitance data for storage into the database and use by the power calculation process.

Buffering strategies for global interconnect were enhanced during the POWER7 design and were used in some unit-level designs. Buffering of signal nets with repeaters is necessary to maintain signal integrity (rise time or slew) before it reaches a receiving latch. Slew targets are maintained and checked per clock phase on each signal net. In previous buffering schemes, only phase tag and slew target information was available when buffering a signal; thus, although a signal slew was achieved, frequency may or may not have been achieved, requiring frequency learning from a subsequent timing run. With the use of the timing-driven BuffOpt tool, units had the ability to buffer and incrementally retune in a single timing run, until the frequency and slew goals were met, thus reducing the number of multiple design cycles. In addition, to assist with the power constraints on the unit, BuffOpt was able to go through a power-down sequence, relaxing signal and path frequencies

in order to achieve power goals as well. Buffer power was reduced by 25% in parts of the POWER7 design using this method.

Not all signal nets needed to be timed in a chip design due to their function in the hardware. These untimed signals typically need to meet a (much slower) slew requirement for noise rejection, but have no frequency requirement. The difficulty and exposure typically encountered is in validating that a signal net that was assumed untimed for timing analysis is assumed at the same state for design verification. POWER7 introduced a novel approach to eliminate this exposure by requiring all untimed signal nets to maintain a very-high-speed integrated circuit hardware description language (VHDL) timing type attribute of UNTIMED. This information was then passed to the verification and timing processes for checking. The design netlist for timing, which retained these attributes, was translated during timing analysis with a signal phase rename such that it was checked for a slew requirement to satisfy noise analysis, but not a frequency requirement. Similarly, for logic built-in self-test verification, an Xstate (i.e., unknown state) injection is performed on untimed nets and checked.

Through these many improvements to existing global timing methodology, the design of the POWER7 chip achieved more rapid and easier timing closure, which will benefit all future IBM microprocessor designs as well.

Design efficiency

A key methodology focus for POWER7 was to enable higher design efficiency. Custom macros heavily leveraged layout automation tools. Growable systems based on Skill** (Cadence Design Systems' scripting language) created many of the chip's arrays and register files. Most importantly, a next-generation synthesis methodology allowed random logic macros (RLMs) to implement the highest performance logic with minimal designer intervention.

Layout automation

Increased use of the layout automation productivity (LAP) tool played a crucial role in custom macro design for POWER7 using four steps.

- 1) *Placement-by-instance parameters (PIPs)*—PIP adds parameters to the gates that define relative placement of each instance for horizontal and vertical positions. These parameters include bit location in x and y planes, rotation of the gates, and polycrystalline layer (poly) skip in the y plane.
- 2) *Placed layout and routing*—LAP generated a physically placed layout for the macro with full connectivity information. These layouts could be imported into the Cadence Virtuoso** Layout XL tool for manual routings or other automated routers. LAP also had a capability to preroute any critical nets on any desired layers.

In POWER7, designers prerouted or shielded critical nets such as clocks and MUX select using the manufacturing-aware router (MAR), which was called from the LAP graphical user interface. LAP had the capability to create a full hierarchical layout at all levels, including "bubble-up pins" from lower levels, to match the schematics or flat layout as needed to serve the purpose of better routings, and to tune for power optimization.

- 3) *Macro pins (I/O)*—LAP enabled schematics designers to place design rule checks (DRCs) and methodology clean pins (I/O) very easily. LAP also provided several choices of where to place and design pins such as on the driving gates, on a particular bit, or at the edge of the macro for better routability. This methodology in LAP not only reduced the layout time but also enabled accurate timing at the macro interface. LAP also had an option to design power, ground, and redundant interconnect vias for the macros.
- 4) *Placed very large scale integration (VLSI) model*—Once macro placement and pin designs were done, the LAP option of calling the Steiner estimated parasitic tool enabled designers to generate a netlist of the design, including estimated parasitic information. These estimated parasitics enabled designers to perform accurate early physical design, timing, electrical, and power checks. This accurate early analysis reduced the need for postlayout iterations and sped timing and electrical closure.

A typical LAP-based custom design flow is shown in **Figure 4**.

Growable arrays

Traditionally, SRAM arrays are designed using a full custom approach. Every transistor is tailored for its specific application. Typically, the effort is on the order of one to two person-years per array. While ever increasing chip functionality drives increases in the number of unique arrays per chip, pressure on cost, and thus resources, does not allow for the staffing of development teams to keep pace. Therefore, more and more arrays are built in an automated code-based methodology, which is called *compilable* or *growable*. Using this approach, the productivity could be multiplied by more than 30, delivering as many as 50 arrays for less than three person-years.

Three distinct code-based array assembly systems were developed to create the register files and arrays of POWER7. The growable register file system delivered 40 unique designs based on ground rule clean cells with up to three read and three write ports, supported key power-saving features such as word-line headers, tristate bit-line drivers, and superhigh threshold voltage devices, and also supported asynchronous and voltage domain crossings between the write and read sides of the design. The growable

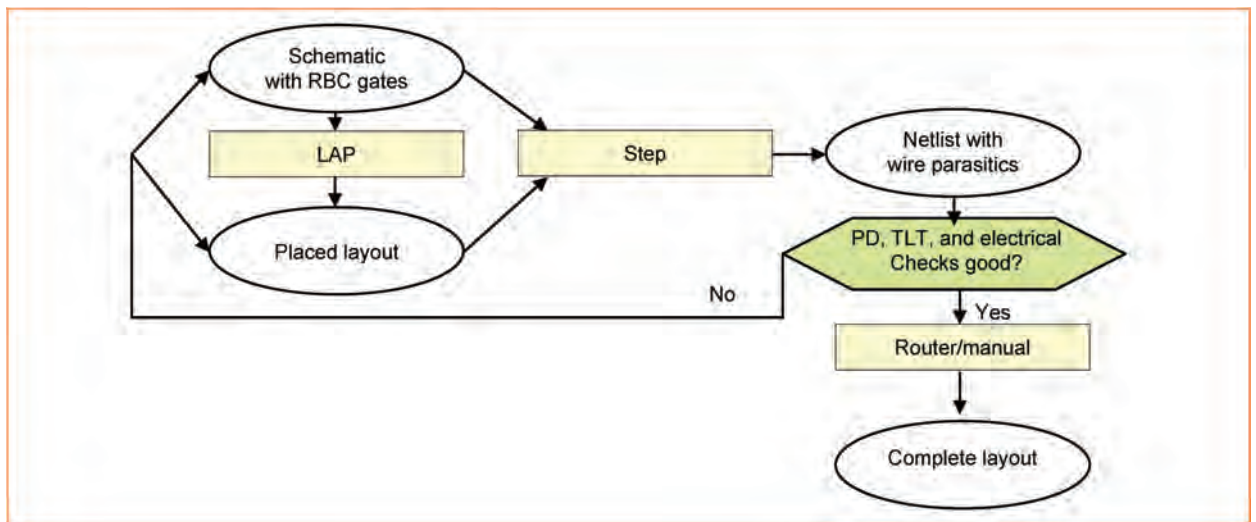


Figure 4

LAP flow for custom macro design. (PD: physical design; TLT: transistor-level timer.)

six-transistor system produced four designs with a single read/write port while using the V_{dd} and V_{CS} rails to support the full core frequency and voltage range. The next section describes in detail the growable eight-transistor (G8T) system that produced 12 unique designs, supported cross voltage and frequency domain designs, and used a $0.635\text{-}\mu\text{m}^2$ cell.

G8T array growable system

We chose the design point for the G8T SRAM arrays (or “G8Ts”) to be an alternative for small register files in nest applications. The 8T SRAM cell allows independent read and write ports while offering higher area density over standard register file cells. They were originally targeted for nest frequency, but certain configurations of G8Ts also meet the core cycle-time requirements and are used in high-frequency clock domains.

The application space described above drove the number and type of options that are offered. To cover a good share of typical register file-use cases, the G8T system offers one write port and up to two read ports. All of these ports are independent and can be hooked up to different clock domains. This flexibility also extends to voltage domains. A G8T can be used in chip areas where only a single supply voltage is distributed. In areas where an additional array power rail with higher voltage is present, a G8T can take advantage of it and operate with increased performance. To increase yield on small processing nodes, the G8T system can build arrays that can be repaired postproduction if they are affected by manufacturing defects. The smallest possible G8T configuration can store 16×8 bits, whereas the largest configuration can store up to 128×144 bits.

The inherent challenge of array compilers is to ensure functionality across all configuration permutations and process variations. If the design and verification effort of a simple array is of the order $O(N)$ and the time to design closure is T , this effort will increase to $O(m! \cdot N)$, with m as the number of configuration options, as every option needs to be functionally verified across the other configurations. It is prohibitive to increase the design time at the same rate to $m! \cdot T$. The answer to that challenge is threefold.

- 1) Good engineering judgment that anticipates critical corner cases and avoids them by choosing a robust design point.
- 2) Automation and parallelization of design build and verification. We developed a growable system that generates VHDL, physical design data, and all other necessary support files ideally with no manual intervention. In addition, we created a tool (Tools Portal) that combines data management with a customizable automated tools flow that fits the G8T system. This ensures consistency and quality in the designs we build.
- 3) Reporting that allows a human designer to quickly grasp the state of the growable system. We developed a tool reporting system that not only has good overview capabilities but also is able to present detailed information in case of tool failures (XML Research Group XML-driven report generator).

RLM synthesis

POWER7 core and nest circuits relied heavily on a novel next-generation synthesis methodology that utilized an extensive multidimensional cell library, including pulsed

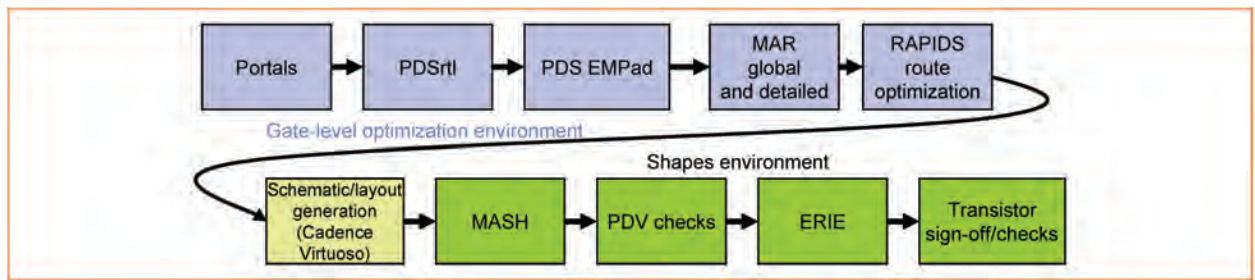


Figure 5

POWER7 RLM design flow (NextGen). (MAR: manufacturing-aware router.)

latches that allowed RLMs to implement the highest performance logic. Stringent electrical constraints, including timing (both late and early mode), transition rate, and clocking constraints, were enforced by synthesis through a *correct-by-construction* methodology that integrated a manufacturing-aware signal and clock routing environment, along with placement-driven synthesis (PDS) [1]. Novel optimizations to implement structured clocking that built horizontal and vertical stacks of latches around clock buffers enabled synthesis to produce robust clocking similar to that of custom designs. A new and novel engineering-change (EC)-order methodology [5] enabled late logic changes to be integrated in the chip efficiently. Overall, this new design closure environment enabled approximately 70% of the logic macros to be automatically synthesized, yielding unprecedented productivity.

Synthesis was extensively leveraged to address errors consisting of two different classes: 1) routing-specific problems resulting from technology insertion (e.g., DRC errors, such as width and spacing violations, or layout-versus-schematic (LVS) effects on standard cell timing rules and parasitics from 1-D extractions of Steiner tools; errors such as opens and shorts required the most effort); and 2) miscorrelation between the optimization and sign-off timing environment (such as timing based on 1-D-extracted Steiner routes in optimization versus that based on a 3-D extracted layout in sign-off). The base router already provided an efficient way to address the DRC and LVS errors in 45 nm and beyond. However, mitigating the miscorrelation problems required an optimization environment supporting timing measurement and correction at a higher level of detail than available in the POWER6* RLM flow [3]. This flow had three distinct environments—a gate-level optimization environment based on IBM's internal gate-level tools running on IBM's internal instance data model, a Cadence Virtuoso library-exchange-format/design-exchange-format-based routing and layout-generation environment, and a layout-based sign-off environment running on a combination of vendor and IBM internal tools.

Of most importance, the sign-off timing flow used a full 3-D extraction of the layout generating a Simulation Program with Integrated Circuit Emphasis netlist timed by IBM's EinsTimer-based transistor-level-timing tool, i.e., EinsTLT. The timing in the gate-level optimization flow used EinsTimer to ensure that the library timing rules, particularly those of the clocking elements [latches and local clock buffers (LCBs)], were robust.

The overall RLM design and synthesis system was called *Next Generation RLM Flow (NextGen)*, as shown in **Figure 5** and discussed in more detail below.

Front-end synthesis with PDSrtl

An integrated environment for design closure known as PDSrtl was created by combining a physical synthesis tool known as PDS (discussed in detail in [1]) with front-end synthesis tools PORTALS and BooleDozer* logic synthesis. PORTALS compiled VHDL into technology-independent gates, and BooleDozer compiled technology-independent gates into a netlist with technology library gates [6]. The technology-mapped netlist is an input into PDS, which is IBM's physical synthesis tool [1]. During the POWER7 design, in addition to timing and routing, stringent electrical checking constraints known as *echs* presented one of the most significant challenges, which were overcome with this new methodology.

The design guidelines specified a set of rules that define permissible and robust circuit implementation guidelines using the cell library. Custom macros and synthesized RLMs are required to honor these design rules to meet the macro sign-off criteria. In the POWER7 RLM methodology, a transistor-level checking tool, which is called EinsCheck, verified these design rules on an extracted netlist. The EinsCheck rules enforced in the RLM flow are broadly categorized into structural, topological, electrical, and physical types with synthesis transforms integrated into the PDSrtl and Routing-Aware Physically Incremental Driven Synthesis (RAPIDS) flows to resolve these design rule violations. The rules impose localized constraints on the

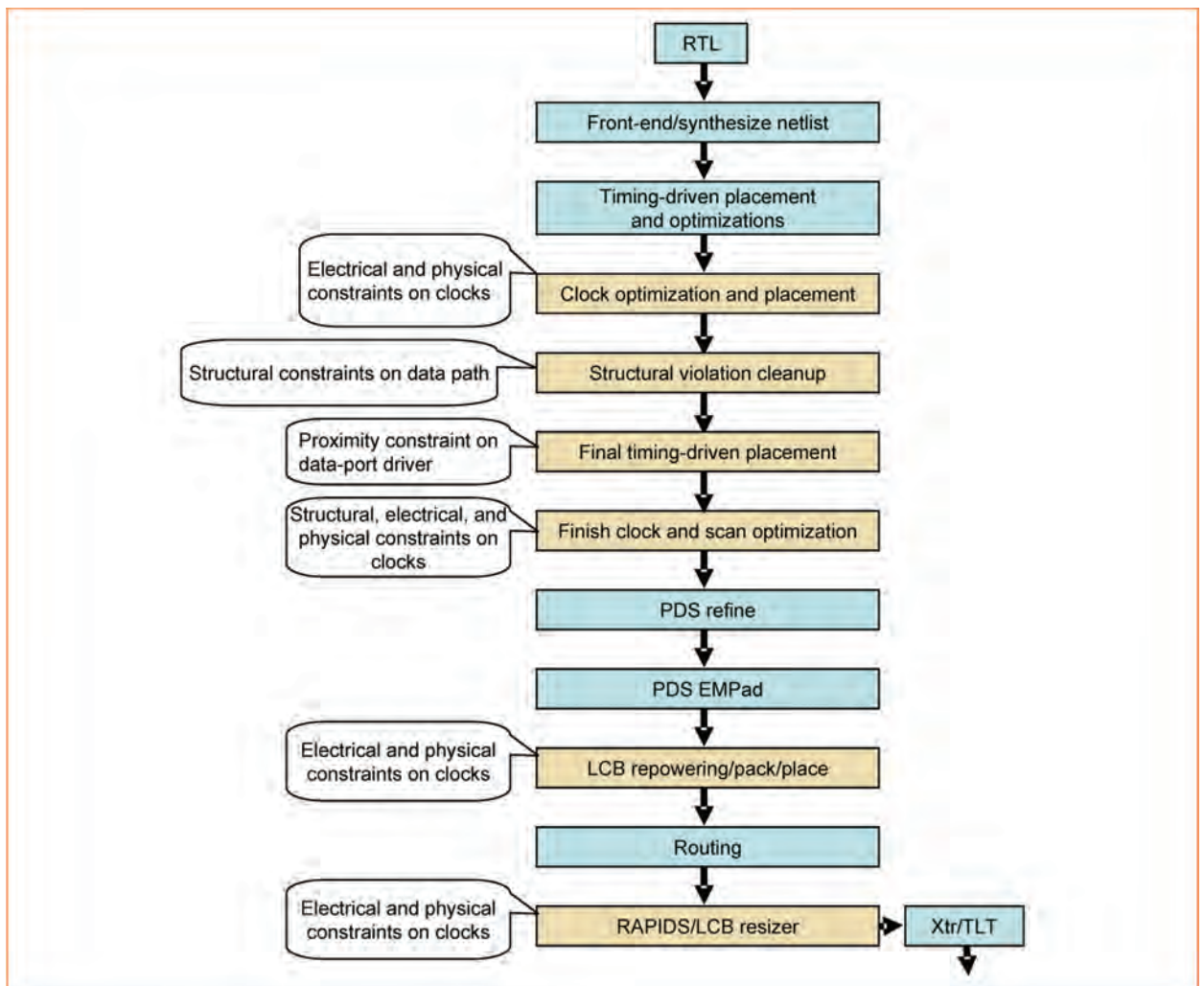


Figure 6

EinsCheck design rule violation cleanup in the RLM synthesis flow. (RTL: resistor-transistor logic.)

gate-level implementation, which mostly pertain to the data and clock signals associated with the latches and LCBs, as well as the control signals feeding the LCB blocks. These EinsCheck constraints are honored through different phases of the RLM implementation flow with appropriate transforms, as shown in **Figure 6**.

The structural EinsCheck rules impose restrictions on the logic gate type and the minimum effective pull-up and pull-down field-effect transistor widths of the cell driving the latch transmission gate data port. Additionally, the gate driving the latch data port is required to have single fan-out (topological rule) and be placed in proximity to the latch, specified by distance limit (physical rule), in order to mitigate noise effects. These structural and topological constraints are satisfied during the *structural violation cleanup* phase in

PDSrtl. The latch and its data-port driver are grouped as a single entity or attracted through weighted virtual nets during the *final timing-driven placement step* to meet the tight proximity constraint. The electrical EinsCheck rules verify the load and resistive-capacitance *RC* delay on the local clock nets from the LCB to latches and the internal LCB components. The *clock optimization and placement step* in PDSrtl builds the local clock tree by clustering latches, cloning the LCBs (as a hierarchical entity), and placing the latch-LCB clusters in order to satisfy the load and *RC* delay limits on the clock nets between the driver LCB and the sink latches. The LCB internal components, which are referred to as *atoms*, are repowered to meet the load limit as a postprocess *LCB repowering step* following the constructive clock optimization phase. The resized LCB *atoms* are grouped hierarchically in

order to form a single compact entity, preserving the relative configuration of *atoms* for each LCB type, and placed in order to reduce the wire capacitance between the *atoms*. The LCBs are also repowered in RAPIDS, which is a postrouting timing closure environment, with more accurate parasitic wire capacitance to resolve the remaining EinsCheck violations. In all, several constructive methods to resolve EinsCheck violations are integrated into the RLM flow to minimize the manual effort required to fix such violations.

Structured clocking

Clock synthesis includes placement of clocked cells (e.g., registers, latches, and flip-flops), clock buffers, and clock gates, as well as routing of the clock distribution network. In high-performance microprocessor design, clock synthesis is one of the most important steps because it directly determines the maximum clock frequency. It is critical to the design quality for three reasons.

- 1) Latch and flip-flop placement dominates the minimum achievable signal delay, which is a major component of minimum clock period.
- 2) Clock buffers and latch or flip-flop placement directly affect the local clock route, which is the major consumer of total clock power dissipation.
- 3) Clock gates and buffers are usually large in size because they are designed to reduce clock skew in clock distribution and they consist of special components for clock gating and splitting.

To minimize clock skew with the existence of variations, the microprocessor usually utilizes clock grids or other regular global clock structures. As a result, macro clock pins often form regular patterns, and if local clock synthesis can take advantage of such regularities, it is possible to further improve timing and reduce local clock power. In **Figure 7**, the yellow horizontal wires are pins of the clock routing structures from higher in the hierarchy. The green boxes are clock buffers, which are driven directly by the clock pins. If one can place the buffers closer to the pins and form horizontal structures, clock power can be reduced.

It is worth noting the difference between clocking structures in high-performance microprocessors such as POWER7 and application-specific integrated circuits. Because of the stringent requirement on clock skew and signal quality in POWER7, latches need to be placed next to their driving LCB, and they need to be placed in a structured way to allow robust clock routing. There are no commercial electronic design automation tools for this task, and in past microprocessors, this has only been achieved by custom design. To the authors' best knowledge, POWER7 is the first microprocessor where regular clocking structures are automatically synthesized.

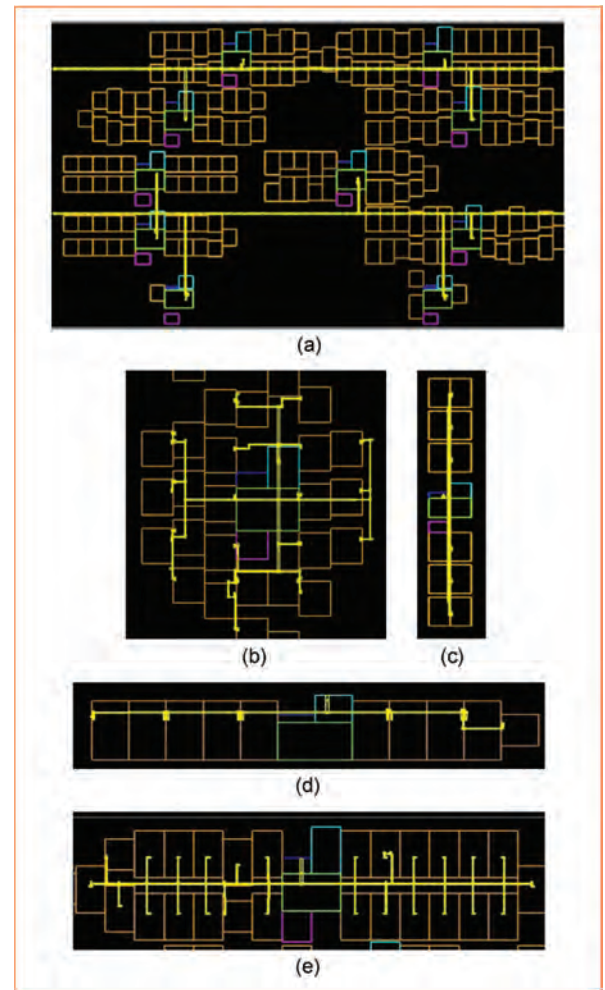


Figure 7

Clock routing topology of structured RLM clocks. (a) RLM with structured latches. (b) Latch huddle clock topology. (c) Single-stack vertical structured latches. (d) Single-stack horizontal structured latches. (e) Double-stack horizontal structured latches.

Structured LCB placement

LCBs are specially designed to repower the clock signal from the higher clock hierarchy to drive the last stage of clock networks, such as latch and flip-flops. They can usually save clock power with local clock gating. The task of the LCB placement step is to determine the locations of LCBs and the latch groups driven by each LCB, whereas the objectives are to minimize the latch-placement distortion when latches are pulled toward the LCBs and to place LCBs as close as possible to macro clock pins, under constraints such as LCB fan-out and latency limits and blockages. For example, given a particular LCB placement solution, a cost function can be defined as the sum of distances that each latch would need to move in placement if this solution were applied. The proposed

solution is to divide the LCB placement task into two subproblems and solve them iteratively.

The subproblem of latch-to-LCB clustering is a standard bipartite matching problem and can be solved either as a min-cost-max-flow formulation or as a linear program. The subproblem of placing LCBs is essentially a 2-D legalization problem with large and nonrectangular objects, each of which is one LCB plus the structured latch stacks that it drives and recalls that there exist preferred LCB locations under macro clock pins. Specialized legalization heuristics are used to solve this subproblem.

Although the solution within each subproblem can be globally optimal, the overall iterative solution is a greedy procedure and stops at the first local optimum that it encounters. To reduce such suboptimality, multiple runs of the iterative algorithm are conducted with randomized initial-guess solutions, and the best of the multiple outcomes is selected as the final solution.

Structured latch placement

If we place the latches as structures, we can route the clock connection with spines of thick metal, which can reduce local clock skew with fewer routing congestion problems left for data paths. As a result, it is desirable to place the latches together with the LCB as structures. Three kinds of structures are shown as examples in Figure 7(b)–(e). The green, pink, blue, and cyan are LCB components. As shown in the figure, the routing is much simpler with latch structures, whereas clock skew is easily controllable. The routing can be done with a thick vertical or horizontal trunk, with spine (or “fishbone”) routing style.

After determining the LCB placement, the structured latch placement problem is simply to find the best location for each latch in the user-specified latch structures and simultaneously minimize the timing degradation due to latch displacement. This is also a bipartite matching problem and can be solved either as a min-cost-max-flow formulation or as a linear program. We build a flow graph with edges created only between 1) the source node and each latch, 2) each latch and each structured latch location, and 3) each structured latch location and the sink node, whereas the cost on an edge is the Manhattan distance between the latch and the structured latch location. Based on this formulation, we can ensure that the optimal clock-routing-unaware timing-driven placement can be maintained as much as possible when the latch structures are formed.

Latch structure selection

In the previous sections, it is assumed that the latch structures are given or specified by designers. However, design productivity can be highly improved if the synthesis tool is able to determine the best latch structure based on a number of criteria, such as macro size, shape, or latch or flip-flop characteristics. Obviously, a “tall” latch structure cannot fit

into a “short” macro area. Despite the macro size and shape, the designers may have a preference for the latch structures. One can easily observe that the “tall” latch structure causes less routing congestion than the single-row structure, although both of them are better than the double-row structure. The user can provide a sorted list of structures, according to his preference, and the synthesis tool can try the structures in the list one by one and with different LCB fan-out limit settings. Using the algorithms presented in the previous two sections, the tool can quickly find the fitted structure that is relatively favorable to the designers.

PDS EMPad

The automated repair of hold violations for the POWER7 microprocessor is performed by the PDS early-mode padding (PDS EMPad) tool. The tool is run as its own step in the methodology, on an RLM in which setup timing has been optimized (by PDSrtl) but before full detailed routing by the MAR. At this stage of the design flow, it is important to minimize any degradation to the critical slack because late-mode timing has already been optimized, and because designers vary the parameters of PDSrtl to help close timing based on routing and transistor-level timing (TLT) results. To achieve this goal, PDS EMPad initializes a dual-analysis-mode timing environment in which best-case and nominal chip timing are modeled simultaneously. This allows the optimizations to insert sufficient delay to fix fast paths without degrading late-mode timing below user thresholds. The EMPad algorithm also utilizes specialized synthesis-placement services to insert buffers or inverters at exact placement locations in the chip image (also known as iMap functionality), negating the need for placement legalization, which could have deleterious effects on setup slack. While the bulk of hold timing correction is performed with the insertion of logic, PDS EMPad contains a suite of early-mode optimizations to help fix problems. These optimizations include repowering, pin swapping, pin moving, and box moving. The deployment of these optimizations in the flow has been tuned for maximum efficacy.

Manufacturing-aware router

The MAR used for the POWER7 microprocessor RLMs is a product of collaboration between IBM and Cadence and is known commercially as Cadence Space-Based Router [7]. Fundamentally, MAR is a three-stage router (global coarse grid path/layer assignment, conduit/track assignment, and detail routing), with additional programmable components for DRC cleanup, design-for-manufacturing (DFM) enhancements, and specialty routing. At a low level, MAR is a shape-based router, whose database is OpenAccess. The object-oriented component architecture of this router has allowed IBM to develop specialized routing flows and applications. Specifically for the POWER7 microprocessor, we designed a flow that satisfies various requirements for

high-performance highly automated microprocessor RLM design. These requirements include an automated specialized clock routing flow, DFM considerations such as redundant vias during clock and signal routing, final DRC and LVS quality specifications, and an automated postrouting timing violation fix-up using incremental integrated routing.

The POWER7 microprocessor clocking strategy is implemented using a global clock mesh. The high frequency of the POWER7 clock requires that a very short and lightly loaded path exist between the clock mesh and the latches. This is implemented by two stages of LCBs between the clock mesh and the latches, and each of these stages requires specialized clock routing. In addition, this clock mesh architecture is carefully distributed across the hierarchy. At the RLM level of hierarchy, three specialized connections were implemented: the *nclk* connection, which is closest to the mesh and connects to a splitter; the splitter-to-LCB connection; and the net connecting the LCB to the latch clock inputs. Part of this architecture requires clusters of latches around the LCBs, which may be structured or unstructured. Latches were designed to be routed in specific patterns that match with the structured latch placements; therefore, a special clock routing algorithm was implemented. Unstructured latch clusters also required a special clock routing topology known as a plus, or crossed spine, topology. All of these special routing techniques were developed to control the delay of the clock signals at each stage, thus limiting the skew of the microprocessor clocks. In addition, because of their high duty rate, the routing of the clocks uses robust or (redundant) vias exclusively.

In the signal routing domain, MAR enabled the definition of a flow that achieved several difficult goals: Primarily, the quality of the router measured in terms of end-to-end DRC and LVS violations was expected to be very high. The tools are expected to produce designs that pass each checking step at the 90th percentile or above. At the same time, the length of individual nets is expected to be close to ideal to minimize chances of timing violations. Various solutions to the problem of scenic routing were implemented, including prerouting, timing-based prioritization, and rip-up and reroute. Finally, DFM requires that the router maximize the use of a “robust” double-cut via that reduces routability. Telescopic wiring was employed from the LCBs to the latch sink connections to optimize resource utilization and delay while satisfying electromigration and power requirements. All of these requirements were met by two elements: First, a highly customized flow was implemented, which contained aggressive global routing targets, robust via optimization, and multiple passes of DRC fix-up. This flow supported customizable backfill, pattern density, decoupling, and well tie placement, and automatic EC of logic elements, as well as frozen mask layer wiring. Second, a novel timing optimization flow that combines buffer insertion, logic optimization, retiming, and routing, called RAPIDS, was implemented.

RAPIDS postrouting optimization

The process of timing and electrical closure of the design after wiring is a challenging process. The approach used in POWER7 integrates the logical, physical, and wiring subsystems together in a single tightly coupled program that solves the timing, electrical, and wiring problems. This program is called RAPIDS. It is constructed to fix transition time and capacitive loading violations, as well as timing failures. Starting with a fully routed design (which may have some opens, shorts, and routing DRC violations), RAPIDS completes with a fully routed design providing timing and routing closure. Shorts and opens are resolved between detailed wires, and routing design rule violations are corrected.

The program begins by reading the output from the previous MAR step (the OpenAccess routed cell view) and timing assertions. Routing-Based Extraction (RBE 2.5D) and delay (RICE) calculators are installed to provide information back to the EinsTimer timing subsystem. Timing is executed in an incrementally changed lazy-evaluated manner for efficient recomputation of dynamic netlist changes. Wires are extracted with a shape-based extraction process and modeled with RICE. Transition effects are propagated throughout the timing graph to provide a high degree of accuracy. The netlist is transformed, placed, and routed simultaneously to accurately model the effect of new routing, physical placement, and orientations of devices. A combination of global routing and detailed routing is employed.

RAPIDS can be envisioned as having two major stages. The first stage is the optimization stage, and the second stage is the EC routing stage.

In the first stage, the program changes the netlist by resizing cells in place or moving to accommodate existing cell resources, inserting buffers along wires into unoccupied space, moving cells to reduce wires to unoccupied space, lowering the threshold voltage of devices, and rebuilding buffer trees. Placement of any new cells or moved cells is performed with a “hole-finding” technique that legally places and orients the cell incrementally during the evaluation of the transformation. Rerouting of any changed nets or pin positions is also performed. Placement is also performed taking into account wiring by limiting locations at which shorts would arise. In addition, contained in this stage is the load minimization of LCB driving nets.

In its second stage, a timing-aware EC router is employed, which determines the conflicting nets from newly shorted wire and pin shapes and reroutes these nets along with any exposed open nets. Nets are weighted by a combined slack and slew criticality metric to guide the detail router on rip-up and reroutes. Design rule violations are resolved through shape manipulation and partial rerouting of nets.

Finally, the RLM is then taken through RLMBuild, which is a Cadence Virtuoso-based layout generation process, and processed through MASH, which is a tool for DFM

layout optimization. Finally, sign-off checks such as timing, DRC, LVS, power/signal electromigration, glitch noise, and other electrical checks are executed on the RLM to ensure a robust design that meets all specifications.

A traditional EC-order process, which is similar to that in POWER6, was used to implement late logic changes in both RIT-A (front-end gates changes allowed) and RIT-B (wiring only changes allowed) modes. A novel RIT-A and RIT-B EC-order process, which is called DeltaSyn [5], was put in place later in the POWER7 design to implement EC orders that will significantly benefit future designs. Furthermore, a novel frozen layer rerouting process was employed to automate EC orders that occur after the beginning of mask generation.

Conclusion

The POWER7 methodology innovations in the areas of modularity, timing, design efficiency, and macro synthesis enabled the chip to achieve its frequency and power goals on a significantly accelerated schedule with reduced physical design resources.

*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 Cadence Design Systems or Sony Computer Entertainment Corporation in the United States, other countries, or both.

References

1. L. Trevillyan, D. Kung, R. Puri, L. Reddy, and M. Kazda, "An integrated environment for technology closure of deep-submicron IC designs," *IEEE Des. Test Comput.*, vol. 21, no. 1, pp. 14–22, Jan./Feb. 2004.
2. K. Bard, B. Dewey, M.-T. Hsu, T. Mitchell, K. Moody, V. Rao, R. Rose, J. Soreff, and S. Washburn, "Transistor-level tools for high-end processor custom circuit design at IBM," *Proc. IEEE*, vol. 95, no. 3, pp. 530–554, Mar. 2007.
3. 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. Shepard, H. H. Smith, D. Thomas, P. J. Restle, J. R. Ripley, S. L. Runyon, and P. M. Williams, "IBM POWER6 microprocessor physical design and design methodology," *IBM J. Res. & Dev.*, vol. 51, no. 6, pp. 685–714, Nov. 2007.
4. 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.*, vol. 46, no. 1, pp. 27–51, Jan. 2002.
5. S. Krishnaswamy, H. Ren, N. Modi, and R. Puri, "DeltaSyn: An efficient logic-difference optimizer for ECO synthesis," in *Proc. Int. Conf. Comput.-Aided Des.*, San Jose, CA, 2009, pp. 789–796.
6. L. Stok, D. S. Kung, D. Brand, A. D. Drumm, A. J. Sullivan, L. N. Reddy, N. Hieter, D. J. Geiger, H. H. Chao, and P. J. Osler, "BooleDozer: Logic synthesis for ASICs," *IBM J. Res. & Dev.*, vol. 40, no. 4, pp. 407–430, Jul. 1996.
7. D. Cross, E. Nequist, and L. Scheffer, "A DFM aware, space based router," in *Proc. Int. Symp. Phys. Des.*, Austin, TX, 2007, pp. 171–172.

Received March 3, 2010; accepted for publication April 21, 2010

Joshua Friedrich *IBM Systems and Technology Group, Austin, TX 78758 USA (friedrich@us.ibm.com)*. Mr. Friedrich was the circuit design leader for the POWER6 microprocessor memory subsystem, and after the completion of its first-pass design, he led its circuit characterization, optimizing both frequency and yield. He led the design of several core units in the POWER7 and has been instrumental in power reduction efforts as power lead for POWER7. Mr. Friedrich is currently working as chip circuit lead and has been a driving force behind the tool, design methodology, and technology team for IBM's next-generation microprocessor POWER8*.

Ruchir Puri *IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (ruchir@us.ibm.com)*. Dr. Puri is manager of the synthesis group at IBM Research where he leads synthesis research efforts for IBM's POWER, zSeries*, games processors, and system chips and has been leading efforts for design productivity with large block synthesis and structured synthesis. He is recipient of an SRC (Semiconductor Research Corporation) Outstanding Mentor Award and has been an adjunct professor at Columbia University, New York. Dr. Puri is a member of the IBM Academy of Technology and an IBM Master Inventor, an ACM Distinguished Speaker, an IEEE Distinguished Lecturer, and a Fellow of the IEEE.

Uwe Brandt *IBM Systems and Technology Group, 71032 Boeblingen, Germany (uwe.brandt@de.ibm.com)*. Mr. Brandt is a development engineer in the Compilable Array group in Boeblingen. He received his diploma and master degree in electrical engineering at the University RWTH Aachen in 2004. He joined IBM Research and Development in Boeblingen in 2004 and works on custom logic circuits and arrays for IBM complementary metal-oxide semiconductor (CMOS) microprocessors.

Markus Buehler *IBM Systems and Technology Group, 71032 Boeblingen, Germany (BUEHLER@de.ibm.com)*. Dr. Buehler received his diploma in electrical engineering at the University of Karlsruhe in 1993 and his Ph.D. degree in computer science at the University of Stuttgart in 2000. He works in the Compilable Array team in Boeblingen. He joined IBM Boeblingen in 1999 where he worked in the Physical Design team. Currently, he is responsible for driving commonality between the different compilable array systems.

Jack DiLullo *IBM Systems and Technology Group, Austin, TX 78758 USA (dilullo@us.ibm.com)*. Mr. DiLullo is a Senior Engineer and part of the Integration/Timing team in Austin, Texas. Currently, he works in IBM Electronic Design Automation specializing in global timing methodology for the pSeries*, zSeries, and gaming chips, as well as performing pSeries global chip-timing closure.

Jeremy Hopkins *IBM Systems and Technology Group, Austin, TX 78758 USA (jthopkin@us.ibm.com)*. Mr. Hopkins received his B.S. degree in electrical and computer engineering from the University of Texas, Austin, in 2004, and he joined IBM in Austin, Texas, working in the Physical Design and Integration team for POWER6. Subsequently, he worked in the IBM Electronic Design Automation team working on synthesis and was the ECO (engineering-change order) methodology lead for POWER7. He worked on large block synthesis support and development for various IBM microprocessors.

Mozammel Hossain *IBM Systems and Technology Group, Austin, TX 78758 USA (mozammel@us.ibm.com)*. Mr. Hossain has been with IBM since 2003 and led the circuit design team for various units in the nest of POWER6 and POWER7 and worked with the tool and methodology team to drive better quality. Prior to working with IBM, he held various positions in digital circuit design with AMD, Mentor Graphics, and Hewlett-Packard. He is presently working as nest circuit lead for POWER8. His research interest includes low power and noise immune VLSI circuit design.

Michael Kazda IBM Systems and Technology Group, East Fishkill, NY 12538 USA (kazda@us.ibm.com). Mr. Kazda received his M.S. and B.S. degrees in computer science from Rensselaer Polytechnic Institute, Troy, New York, in 1994 and 1989, respectively. He is a Senior Software Engineer and leader of the RAPIDS project used for post-routing design closure. He holds a patent for the latter work.

Joachim Keinert IBM Systems and Technology Group, 71032 Boeblingen, Germany (KEINERT@de.ibm.com). Mr. Keinert received his M.S. degree in electrical engineering from the Technical University of Stuttgart, Germany, in 1980. He joined IBM in 1979 and has worked on CMOS circuit tool development and chip-design methodologies since 1982. Currently, he is responsible for ECO methodologies for RLMs.

Zahi M. Kurzum IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (kurzum@us.ibm.com). Mr. Kurzum received his B.S. and M.S. degrees in computer engineering from Clemson University in 1984 and 1989, respectively. He is a Senior Engineering Manager at IBM Systems and Technology Group where he manages the development of physical design flows and methodology for RLM microprocessors, ASICs, and gaming designs. He has more than 20 years of experience in electronic design automation at IBM in the area of VLSI physical design.

Douglass Lamb IBM Systems and Technology Group, Research Triangle Park, NC 27709 USA (douglass@us.ibm.com). Dr. Lamb earned B.S.EE/C.S. and M.S.EE degrees from the University of Virginia and an ECE Ph.D. degree from George Mason University in processor modeling. He is the electronic design automation team and routing leader of RLM flow for POWER microprocessors. He holds four patents and is a Professional Engineer.

Alice Lee IBM Systems and Technology Group, Lexington, MA 02421 USA (ahlee@us.ibm.com). Ms. Lee is an Advisory Applications Engineer working in the IBM Electronic Design Automation Group. She has supported physical design, optimization, and logic synthesis stools for IBM ASICs as well as the POWER6 and POWER7 projects.

Frank Musante IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (fmusante@us.ibm.com). Mr. Musante received his B.S. degree in computer science from New Jersey City University in 1998 and joined IBM Electronic Design Automation team in Fishkill, New York. He works as a Software Engineer in the support and development of placement-driven synthesis and RAPIDS.

Jens Noack IBM Systems and Technology Group, 71032 Boeblingen, Germany (NOACK@de.ibm.com). Mr. Noack leads the compilable 8T SRAM array team in Boeblingen. He received his diploma in electrical engineering at Technical University of Cottbus in 1998 and joined IBM in 2002. He worked on developing compilable SRAMs for the Cell Broadband Engine** processor and the POWER7 microprocessor, and on establishing the compilable SRAM array methodology for IBM mainframes.

Peter J. Osler IBM Systems and Technology Group, Essex Junction, VT 05452 USA (osler@us.ibm.com). Mr. Osler has B.S. and M.S. degrees in electrical engineering, and a B.S. degree in computer science, all from Massachusetts Institute of Technology. He is a Senior Engineer working in the Electronic Design Automation organization. He joined IBM in Yorktown Heights, New York, in 1983. Since 1987, he has been working at IBM's Essex Junction facility, designing microprocessors and ASICs, and developing electronic design automation software. Currently, Mr. Osler focuses on RLM design automation methodology for the pSeries and zSeries.

Stephen Posluszny IBM Systems and Technology Group, Austin, TX 78758 USA (Stephen.Posluszny@us.ibm.com). Mr. Posluszny received his B.S. and M.E. degrees from Rensselaer Polytechnic Institute in 1982. He is a Senior Technical Staff Member at IBM and currently leads RLM design execution for the POWER7 and POWER8 microprocessors. His research interests include timing closure and design methodology for complex, high-performance microprocessors.

Haifeng Qian IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (qianhaifeng@us.ibm.com). Dr. Qian received his Ph.D. degree from the University of Minnesota in 2006 and since then has been a Research Staff Member at IBM Research where he works on high-performance clock synthesis for IBM servers. He received an ACM SIGDA best Ph.D. Dissertation Award and a Best Paper Award at DAC 2003.

Shyam Ramji IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (ramji@us.ibm.com). Dr. Ramji holds a B.S. degree in electrical engineering from National Institute of Technology, Tiruchirappalli, India, and M.S. and Ph.D. degrees in computer engineering from Syracuse University. He is a Senior Engineer and team lead for the circuit placement tool, CPLACE. He led the development efforts for the placement-driven synthesis tool used in the POWER6 microprocessor design.

Vasant Rao IBM Systems and Technology Group, Hopewell Junction, NY 12533 USA (raov@us.ibm.com). Dr. Rao received the B.Tech. degree in electronics engineering from IIT Madras, India, and M.S. and Ph.D. degrees in electrical engineering from University of Illinois, Urbana. He currently leads the transistor-level static timing analysis tools team within electronic design automation at IBM. He has coauthored several journal and conference papers and holds several patents.

Lakshmi N. Reddy IBM Systems and Technology Group, Yorktown Heights, NY 10598 USA (reddyl@us.ibm.com). Dr. Reddy received a Ph.D. degree in electrical and computer engineering from the University of Iowa, Iowa City, in 1992. Currently, he is a Senior Engineer at IBM in the Electronic Design Automation Laboratory in Yorktown Heights, New York, where he works on synthesis optimization technology for IBM chip designs.

Haoxing Ren IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (haoxing@us.ibm.com). Dr. Ren received his B.S. and M.Eng. degrees in electrical engineering from Shanghai Jiao Tong University, in 1996 and 1999, respectively, an M.S. degree in computer engineering from Rensselaer Polytechnic Institute in 2000, and a Ph.D. degree in computer engineering at University of Texas at Austin in 2006. He is a Research Staff Member at the IBM T. J. Watson Research Center. His research interest includes logic synthesis and physical design for high-performance circuits.

Thomas Rosser IBM Systems and Technology Group, Austin, TX 78758 USA (rosser@us.ibm.com). Mr. Rosser received his B.S. degree in electrical engineering at the University of Missouri at Columbia in 1976. He is an IBM Senior Technical Staff Member. In his 30 years, he has worked on almost all aspects of design automation tools at IBM. He currently leads the RLM flow for all IBM processors.

Benjamin R. Russell IBM Systems and Technology Group, Waltham, MA 02451 USA (benrruss@us.ibm.com). Mr. Russell graduated with a B.S. degree in computer systems engineering in 2006 from Rensselaer Polytechnic Institute, Troy, New York, and joined the IBM Electronic Design Automation team in 2007. He has worked on standard cell timing rule generation and timing correlation for the

IBM POWER7 and ZGryphon processor designs. He is currently working on gate-level sign-off methodology for the POWER8 processor and future microprocessors.

Cliff Sze *IBM Research Division, Austin Research Laboratory, Austin, TX 78758 USA (csze@us.ibm.com).* Dr. Sze received the B.Eng. and M.Phil. degrees from the Department of Computer Science and Engineering at the Chinese University of Hong Kong and the Ph.D. degree in computer engineering from the Department of Electrical Engineering, Texas A&M University. He is a Research Staff Member working for IBM since 2005. He is currently a Senior Member of the IEEE.

Gustavo Téllez *IBM Systems and Technology Group, Essex Junction, VT 05452 USA (tellez@us.ibm.com).* Dr. Téllez received B.S. and M.S. degrees in electrical engineering from Rensselaer Polytechnic Institute, Troy, New York, in 1985, and a Ph.D. degree in computer science from Northwestern University, Evanston, Illinois, in 1996. He is a Senior Technical Staff Member working for IBM since 1986. He is currently working on several chip routing areas for the IBM microprocessor and ASIC design teams.

Functional verification of the IBM POWER7 microprocessor and POWER7 multiprocessor systems

K.-D. Schubert
W. Roesner
J. M. Ludden
J. Jackson
J. Buchert
V. Paruthi
M. Behm
A. Ziv
J. Schumann
C. Meissner
J. Koesters
J. Hsu
B. Brock

This paper describes the methods and techniques used to verify the POWER7® microprocessor and systems. A simple linear extension of the methodology used for POWER4®, POWER5®, and POWER6® was not possible given the aggressive design point and schedule of the POWER7 project. In addition to the sheer complexity of verifying an eight-core processor chip with scalability to 32 sockets, central challenges came from the four-way simultaneous multithreading processor core, a modular implementation structure with heavy use of asynchronous interfaces, aggressive memory subsystem design with numerous new reliability, availability, and serviceability (RAS) advances, and new power management and RAS mechanisms across the chip and the system. Key aspects of the successful verification project include a systematic application of IBM's random-constrained unit verification, unprecedented use of formal verification, thread-scaling support in core verification, and a consistent use of functional coverage across all verification disciplines. Functional coverage instrumentation, which is combined with the use of the newest IBM hardware simulation accelerator platform, enabled coverage-driven development of postsilicon exercisers in preparation of bring-up, a foundation for the desired systematic linkage of presilicon and postsilicon verification. RAS and power management verification also required new approaches, extending these disciplines to span all the way from the unit level to the end-to-end scenarios using the hardware accelerators.

Introduction

From the outset, the POWER7* verification approach aimed to achieve the overall goal to build a high-quality chip on an aggressive schedule and address the verification challenges of a new processor design, which dramatically stretched functional complexity in many dimensions. This directly forced the need to shape methodology and execution to provide unprecedented quality for first-silicon functionality.

The schedule challenge resulted in a plan to completely skip a whole silicon tape-out cycle compared with prior

projects. This was an integral part of a larger accelerated system development plan that was coined “shift left,” illustrating significant compression of the overall schedule.

The POWER7 microprocessor chip, which is described in detail in another paper [1] in this issue, combines eight processor cores. This quadruples the number of cores compared with the POWER6* processor. Each core supports four-way simultaneous multithreading (SMT4). For the first time, a shared on-chip Level 3 (L3) cache, which is based on embedded dynamic random access memory (DRAM) cells instead of static RAM, is integrated on the chip. In addition to the eight cores, the chip (567 mm²,

Digital Object Identifier: 10.1147/JRD.2011.2117370

©Copyright 2011 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied by any means or distributed royalty free without further permission by computer-based and other information-service systems. Permission to republish any other portion of this paper must be obtained from the Editor.

0018-8646/11/\$5.00 © 2011 IBM

1.2 billion transistors) has a private Level 2 (L2) per core, the shared L3, up to seven ports for symmetric multiprocessor connectivity, and two memory controllers that can transfer 100 GB/s of data. Furthermore, additional and new functions for reliability, availability, and serviceability (RAS) and power management [2] have been placed on the chip together with multiple phase-locked loops and various asynchronous interfaces between the components of POWER7. This breadth of design complexity made the verification challenge manifold.

This paper describes the key initiatives taken to successfully address the above goals and challenges. The focus is on areas that differentiate the POWER7 project from prior projects, namely, POWER4*, POWER5*, and z10*, which are described in [3–5]. As in these prior projects, the POWER7 verification process follows the same hierarchical buildup from the unit level to the system level, as described in [3]. We begin by describing the unit-level constrained-random methodology. We describe the base toolset and then discuss its application for the example of the memory controller unit, which required one of the most complex unit verification efforts. Crucial technology advances in the toolset for formal verification (FV) and semi-FV allowed for a step function of more integrated and broader usage of these technologies.

The new POWER7 core with its SMT4 support required significant advancements in core-level verification, exceeding what was described in [3, 4]. The “Core verification” section also shows the strength of the expansion of verification coverage across many levels of the verification process.

A key new initiative of POWER7 verification was a strong linkage between presilicon and postsilicon verification, which was exemplified in the exerciser-on-accelerator (EoA) effort. Through this activity, postsilicon methods were applied to the presilicon development phase. At the same time, we enabled presilicon verification coverage methods to be applied to measure the effectiveness of the postsilicon validation technology even before tape-out.

Key design aspects of POWER7 as a server microprocessor are its RAS and power management support. These are functional areas that are also referred to as pervasive design, as they tend to cut across traditional processor unit boundaries. We discuss how the methodology for pervasive function verification evolved using new approaches more in line with our mainstream constrained-random methodology. In addition, we discuss some novel structural verification methods that have been used.

This paper concludes with a description of power management verification, which successfully addressed a new design domain spanning low-level power controls to system-level power management strategies.

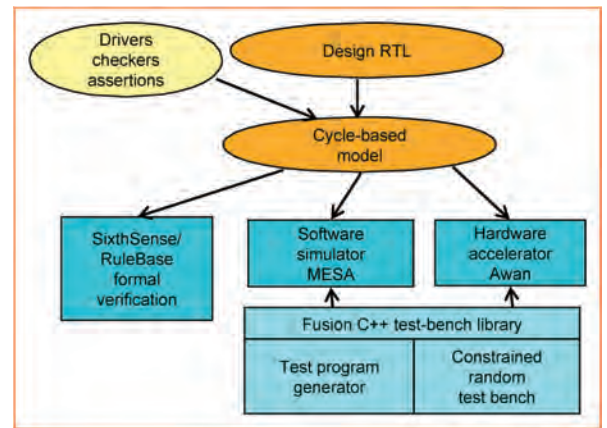


Figure 1

Core methodology flow.

Unit verification

Simulation toolset

Constrained-random techniques for functional unit verification have been developed for IBM server designs for several decades, and there is a large skill base of verification experts in this discipline. For the POWER6 and z10 server processor projects, IBM developed a single common C++-based platform for constrained-random test-bench development [5]. As this platform combined the best techniques and constructs from many prior server projects in a new unified approach, it was called *Fusion*. C++ was used as the base implementation language to guarantee maximum performance scalability, improving at the same rate as C++ compilers on any compute platform. The key differentiator of Fusion to similarly targeted tools in the industry is its runtime and memory efficiency to enable large system simulation and the ease of integration of third-party software into our simulation environments. For POWER7, the Fusion methodology was formalized to better support vertical and horizontal reuse of checkers in hierarchical verification and verification intellectual property (IP) code reuse.

The backbone of all verification engines in simulation, hardware-accelerated simulation, and FV continues to be the cycle-based execution model of the design under test (Figure 1). Having a single consistent interpretation model of the register transfer language (RTL) specification, regardless of which tool is used for a particular verification task, is an invaluable productivity advantage. This way, all stimulus generation, checking, and waveform signatures are portable across the whole tool’s flow.

POWER7 used all of IBM’s cycle-based verification engines, the IBM software-based simulator [Monotonic

Evolutionary Simulation Algorithm (MESA)], hardware accelerators of the asynchronous wide-area network (AWAN) family [3], as well as the FV engines SixthSense [6] and RuleBase [7].

Unit verification relies completely on MESA, i.e., IBM's current cycle-based simulator with an event-driven core engine, which is able to scale software-based simulation all the way from the unit to very large multichip POWER7 models that even include input/output chips.

Base methodology

Unit verification is the workhorse of the verification effort. Random drivers are used to drive traffic into the interfaces of the units under test. The drivers use biased-random constraints and real-time feedback from the unit models to achieve this. The bulk of all checkers used at all levels of simulation are written by engineers working at the unit verification level. The checkers are written in a layered fashion with the intention of being promoted to higher levels of simulation for use there (core, chip, and system levels).

The primary strength of unit verification is direct control in testing. Because the driving stimulus comes directly from the interfaces for the unit, the drivers can explore all areas of the interface's state space equally easily, whereas the real unit connected to it in the full chip will typically explore some of those areas very rarely. There is no filtering effect of other upstream units on the controls for the driver in order to hit the desired traffic streams. Furthermore, the relatively small model size to be simulated (compared with full chips or systems) translates into the highest number of cycles simulated per second. Therefore, unit simulations easily attain orders of magnitude more cycles than all other levels of simulation that are using software simulators.

At the end of the POWER4 project, we developed an extended very high speed integrated circuit (VHSIC) hardware description language (VHDL)-based method for functional coverage and assertion instrumentation. The method, which is called *BugSpray*, is used by the design teams to annotate the RTL with coverage events easily and productively. Thus, a majority of the verification coverage instrumentation is handled by the design team and is intended to let them grade the verification effort from their knowledge of the design implementation. BugSpray was extended greatly for increased productivity for POWER7. BugSpray coverage instrumentation is similar to but simpler than property specification language [8], which can also be used in its place. Part of the toolset is also a coverage data collection database server that is highly scalable to be used in a large simulation workstation farm. All BugSpray coverage events are included at the unit level; for model-size reduction reasons, these are categorized and prioritized so that higher levels of simulation only need to pick up the most important of those events. The combination of more direct control and greater numbers of cycles translates into unit

verification being the environment with the largest success rate in achieving maximal coverage of testing the unit.

The portability of synthesizable coverage and assertion specification between simulation, hardware acceleration, and FV was critical to our approach to cross-link the different verification efforts more effectively. For example, the same basic coverage feedback was used to grade unit verification, chip-level simulation, and the effectiveness of postsilicon exercisers by running them presilicon on the AWAN accelerator. This ensured maximum reuse of functional coverage instrumentation across all verification disciplines and a uniform metric of the effectiveness of each method used.

Unit verification example: Memory controller design

One of the most challenging units to verify in the POWER7 design was the memory controller. With a very tight schedule and a massive increase in design complexity, the memory controller verification demonstrated several of the challenges and pitfalls of unit verification.

The POWER7 memory subsystem experienced a significant increase in complexity in both mainline and RAS function compared with prior designs [1]. The primary responsibility of the host memory controller is to access main memory in an efficient manner, both for performance and power management, while maintaining a double-data-rate three synchronous DRAM interface protocol. Additionally (like the POWER6 memory controller), the POWER7 memory controller maintains a directory, marking lines that have copies off-node, the maintenance of which is critical to proper cache coherency. POWER7 added an arithmetic logic unit so that limited operations could be done directly without the use of a processor core.

The new RAS capabilities include a 64-byte error-correcting code (ECC) with retry and symbol and chip marking, bus error retry, and selective memory mirroring (SMM). The data error detection and correction changed from a single-pass 32-byte ECC algorithm in previous memory controllers to a multipass 64-byte ECC algorithm to detect data errors. The data bus error detection and correction changed from a single-pass ECC algorithm to a cyclic redundancy check (CRC)-with-poison algorithm for detecting and signaling errors on the bus. As errors were detected, the memory controller would then make several passes to retry operations and resolve the operation. In case of failing the retry operation, the memory controller would try to reinitialize the channel while holding up in-flight memory operations before resuming. Additionally, late in the project, SMM was added to the memory controller.

Each of the new features, i.e., 64-byte ECC retry, bus error retry, and mirroring retry, caused an exponential increase in the state space. Because all three functions had interactions with each other, the increases had a multiplicative effect rather than an additive one. As evidence of the increased size of state space, the number of BugSpray coverage events

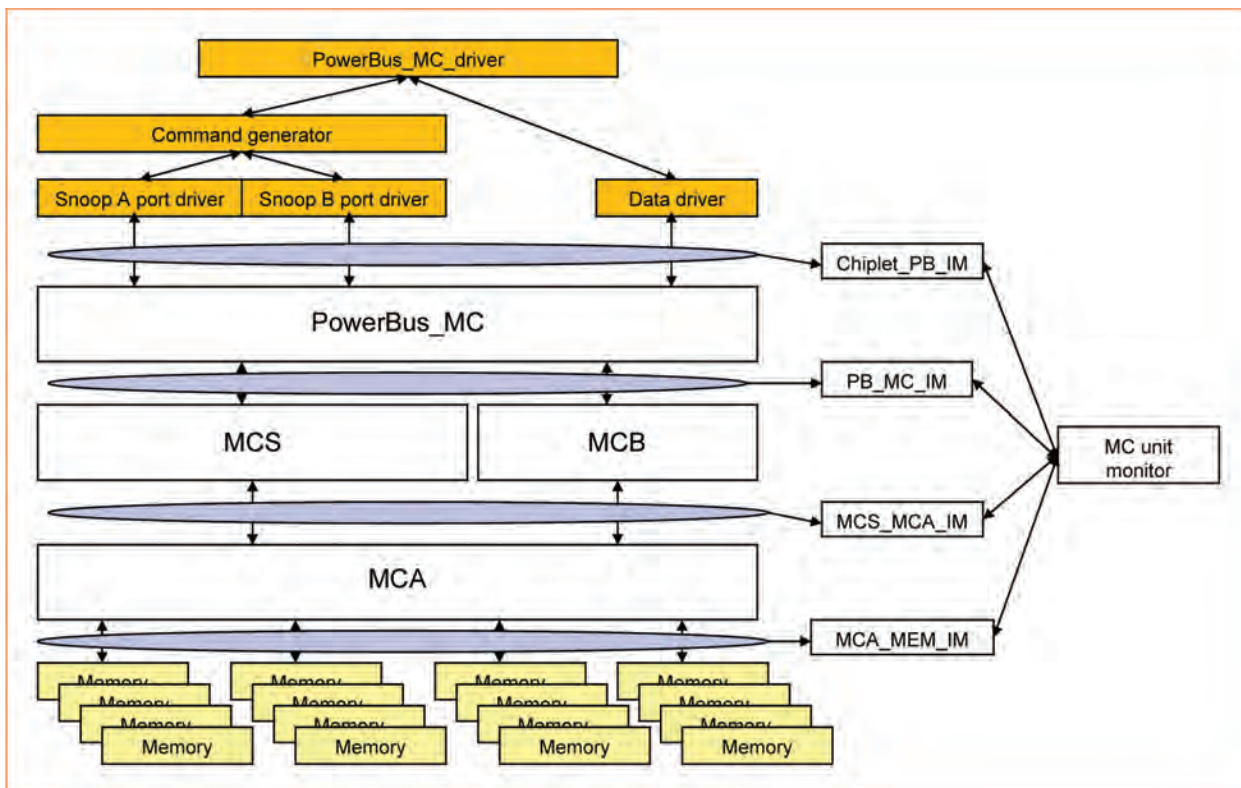


Figure 2
Memory controller (MC) random unit environment. (MC: memory controller; PB: PowerBus; IM: interface monitor; MCS: synchronous part of the MC relative to the PowerBus; MCA: asynchronous part of the MC relative to the PowerBus; MCB: buffer for read/write data.)

inserted in the memory controller to track the state space and ensure sufficient coverage by the verification team increased 10-fold from POWER6 to POWER7.

Memory controller unit environment

Like all of our unit environments, the memory controller unit consisted of the MESA design simulation model constructed from the design VHDL, Fusion C++ drivers and checkers, and BugSpray coverage models (**Figure 2**).

The checkers in the environment were partitioned into interface monitors (IMs) and unit-monitor checkers. The IMs are responsible for checking interface protocols and packaging events into transaction classes for use by other checkers. The unit checkers were written in layers, each with different redundant granularity and serving as a safety net for each other—architectural checks layered across microarchitectural checks layered across implementation-specific checks. The architectural and microarchitectural checks in the memory controller would catch problems with incorrect data being returned to the rest of the system. However, they would not pinpoint the exact point of failure and required more debugging. The

implementation-specific data checks are rules in which the logic designer specifies exactly how a function should work. While deviating from that behavior may not immediately cause a direct architectural or microarchitectural violation (e.g., architectural data violations typically require a read to the location to discover), these checks can most often find bad behavior right at the source and involve the least debug time when they produce failing test cases. However, they also are the checks most often subject to revision, as bug fixes and late-added features can change how the implementation is supposed to work.

As much as possible, unit monitors employ black-box checkers, looking only at well-defined external interfaces for the unit. There were several areas where we were able to follow this convention. However, a few design choices for error recovery required the monitors to look deep into internal signals of the logic. This led to many challenges, as error recovery had the largest increase in complexity. As the project progressed, the amount of information the monitors needed to observe increased. Maintaining cycle accurate copies of internal state was very challenging because the

internal signal timings shifted quite often between logic releases.

All units connecting to the PowerBus [1] share a common interface, which was leveraged to provide a common driver. That driver included generic knowledge about the PowerBus coherency protocol, the command types and contents, as well as a large degree of control for biasing randomness in the command and response packets. System behavior plug-ins were added per unit to further constrain traffic that was relevant to that specific unit. For instance, the memory controller added behaviors to model benchmark workloads by constraining address and transfer type streams. The advantage of shared code comes with the pitfall that users often did not know all of the generic controls of the driver well. This led to some escapes to higher levels of simulation, a tradeoff that is acceptable given the resources that could be saved.

Error injection required paramount attention to cover the state space introduced by the new RAS features. The new multipass ECC algorithm, CRC bus error retry, and SMM all would reissue operations out to the memory dual inline memory modules. Furthermore, all three of those features had notable amounts of cross-coupling in their operation and could interrupt each other's operation to satisfy higher priority errors. Thus, intermittent errors had a much more dramatic impact on the functional operation of the memory controller relative to intermittent errors in previous verification projects.

For the POWER7 memory controller, we saw a dramatic improvement in the quality of our error injection. In previous projects, errors were injected at the start of the test with no regard for what errors were injected in other locations. In this project, the locations of all errors throughout the memory subsystem behind a memory controller had to be coordinated, including the information about whether errors would be injected in a hard fashion or in an intermittent fashion in those locations. An enormous amount of BugSpray coverage events were added to provide feedback. Random error injection (preloaded ECC errors and bus errors) provided substantial coverage. However, cross-coupling ECC and bus errors with SMM required runtime feedback from the memory controller. By using hints observed in the simulation, the error injector was able to target deeper state space.

Like many of the other units on this chip, the memory controller possessed an asynchronous clock boundary, crossing from a clock domain for the PowerBus interconnect to the memory clock domain. This is not new, as previous memory controllers have had asynchronous clock crossings. For POWER7, several improvements have been made for the cross-domain crossing checks, the insertion of skewing logic, and the modeling of the clock signals input to the units, which now have varying periods that average out to noninteger periods. The combination of the

enhancements did help find problems on the asynchronous interfaces much sooner in the project relative to previous projects.

To combat the increase in design complexity and resultant state-space explosion, we heavily increased our internal stimulation on the unit. As much as possible, we tried to preload state machines of the memory controller to shortcut to different starting points. Likewise, we used irritators on internal arbitrators where possible to stretch timing windows and explore states that were theoretically possible but extremely difficult to hit with traffic directly on the interfaces. Despite these actions, many memory controller unit tests ran 10–100 times longer compared with previous projects in an attempt to explore more of the state space. This had a very detrimental effect on debug time. Where many tests would complete in 30 min on POWER6, the analogous tests would take half a day in POWER7.

Formal verification

IBM has a long history of developing robust FV and semi-FV technology [6, 7]. The POWER7 design and FV teams used IBM's FV and semi-FV toolset, which includes *SixthSense* and *RuleBase*. The dramatically increased model capacity of the SixthSense tool allowed FV to span designer verification, sequential equivalence checking (SEC), and deep FV of large design partitions such as the complete floating-point unit (FPU) dataflow.

The mandate coming into POWER7 verification, based on analysis of bugs in past projects, was to apply FV technology extensively to improve presilicon design quality and minimize bug escapes into silicon. The result was the largest and most successful application of FV in any project. The efforts culminated in flushing out hundreds of bugs, many of which would have been extremely difficult to find using traditional verification methods, and ensured correctness of large parts of the logic by way of obtaining proofs. FV application can be best summarized as a combination of an upfront defined methodology and results from deploying the methodology during project execution. The following sections describe the key areas.

Integrated approach

We pursued an integrated approach—positioning FV synergistically alongside simulation and making unit verification teams responsible for defining and owning (project managing) FV based on their respective needs—as opposed to the plan being owned and managed by the FV team as in the past. This allowed for the plan to be dynamically altered, addressing specific requirements and deficiencies in real time, to make FV application more effective. We endeavored to apply FV early on complex logic blocks identified for formal checking in an attempt to provide value-add upfront to complement simulation-based verification. A rigorous process was instituted whereby

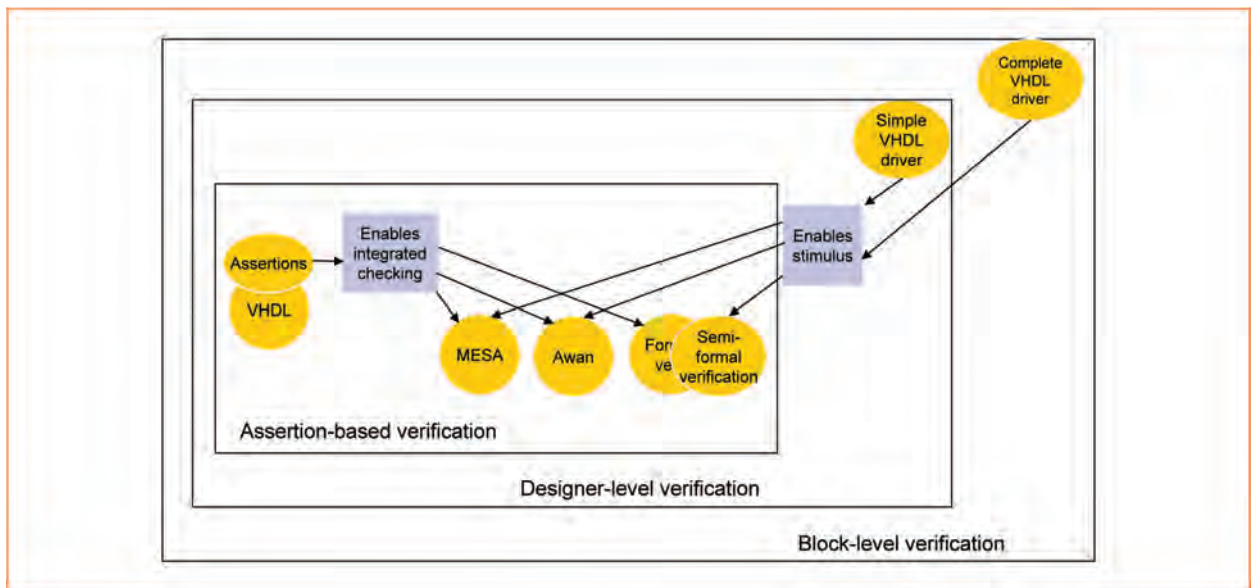


Figure 3

Integrated approach.

block-level environments were reviewed closely with designers, architects, formal experts, and verification engineers to ensure completeness of the checking and correctness of the assumptions. The checkers (properties) and assumptions were added to the simulation environment to be cross-checked and cross-leveraged. We aggressively drove an assertion-based verification paradigm that brought together designer-level verification, FV, and simulation in a unified integrated methodology (Figure 3).

Demonstrated best practices

FV experiences and successes from past projects were heavily leveraged in the POWER7 program. Complex logic blocks on the different units were identified and prioritized for targeted “deep dive” verification by FV experts [3, 9]. The blocks were chosen in consultation with design and verification teams on the basis of their complexity, difficulty in verifying them with other verification disciplines, and logics that have exhibited late bugs on previous projects. Block-level verification ranged from highly intrusive checking of the logic by formal experts based on a deep understanding of the microarchitecture (e.g., prefetch/fetch, issue queue, and memory management), strong FV skills, and past experiences to end-to-end checking to verify conformance of the function against a specification (e.g., queues and ECC), to verification of the algorithmic correctness of the logic [e.g., least recently used (LRUs)]. Application areas included all areas of the chip (core, nest, and pervasive) and hierarchies (blocks, unit, element, and

chip). We pioneered various innovative and reusable techniques and methodologies. Examples include a method to verify correctness and performance of complex arbitration logics by accurately computing request-to-delay bounds and ascertaining the fairness requirements of the arbitration scheme [10], as well as systematic methods to verify complex 64-byte ECC, LRUs, and hardware data structures such as linked lists.

Proven and established methodologies were utilized to verify (complete) function of logics such as FPU dataflow [11] (add, multiply, and fused multiply-add instructions), arithmetic functions (e.g., adder, multiplier, and divider), by comparing the high-performance implementation against a high-level reference model. We pursued applying FV to verify deep and complex pervasive logics [12] building atop past successes and expanded it to include additional areas (e.g., sensor networks), and we created automated methodologies to improve productivity—e.g., Debug Bus verification via creating automatic test benches given a specification in a custom language. In a number of cases, FV was the technology of choice to be relied upon heavily to ensure correctness of oftentimes critical logic (e.g., LRUs, ECC including 64-byte ECC, FPU, and arbitration). The environments created to verify blocks proved to be very useful later to quickly find the root cause of postsilicon problems seen on the laboratory floor. FV has assumed a central role over the years in sorting out bugs in silicon and verifying fixes to the logic.

Sequential equivalence checking

SEC [13] was extensively deployed at different stages of the project to achieve various goals, such as verifying nonfunctional design (e.g., power and timing) changes, verifying external IP conversion over to IBM's clocking and latching methodology, and ensuring mode latches did indeed revert the design back to a previous function. We made this easy-to-use yet powerful technology available to designers to substantially improve productivity by obviating the need to rerun costly regressions to verify nonfunctional changes to the design. In later stages of the project when all functions were completed, we instituted a rigorous end-to-end SEC process starting at the macros and working up to the chip level (with "black-boxing" lower levels of the hierarchies at a particular hierarchy) to establish that inadvertent functional changes did not get introduced in subsequent releases of the RTL.

New exploration areas

We endeavored to push the capacity limits of FV application by exploring several new areas such as creation of constraint-based environments at higher levels of abstraction, for example, at the subunit level, to take advantage of well-defined interfaces and increase the interplay with simulation. The latter is achieved by leveraging formal-to-verify assertions and to weed out unreachable coverage events or obtain hints of how to hit tough coverage events in order to enhance simulation test buckets. This increases state-space coverage with semiformal runs, thus leveraging the superior speed and capacity of IBM's semiformal toolset [6]. We pursued verification of system-level aspects in several contexts. These were, for example, timing protection windows to ensure coherence or deadlock-free operation of the system using direct mathematical analysis on dedicated high-level models. These efforts helped flush out architectural issues early on and assisted in setting up effective simulation test benches to check for violations. As another example, we made attempts to reason about the effect of asynchronous aspects at the system level by modeling these interfaces on higher level models with design details suitably abstracted.

Core verification

Doubling the number of threads per core from POWER6 to POWER7 had a large impact on the verification effort. Aside from the obvious aspects that having more threads in-flight meant more state space to verify, the scalability of test generation was also a factor that needed to be addressed by the verification methodology. As described in [4], the primary means of verifying an SMT core in the past two-threaded designs was to generate independent instruction streams for each thread in much the same manner as had been done for previous single-threaded but multicore designs [3]. However, having to generate four independent

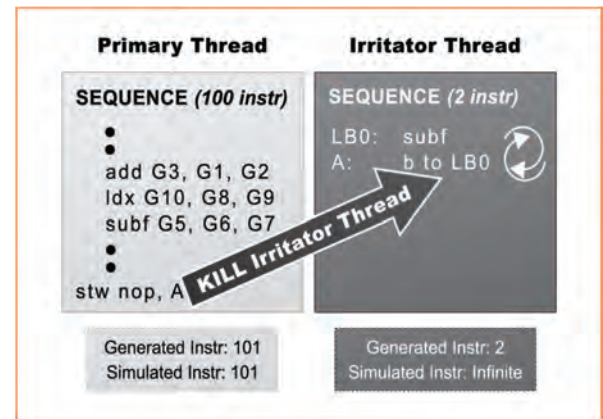


Figure 4

Irritator thread method.

threads for every test case run on a processor core model more than doubled the test generation time. While this method was employed as part of the POWER7 effort, we also needed to create additional test generation methods that not only scaled better but also addressed several shortcomings of the earlier POWER5 approach.

The first new test generation method was developed near the end of the POWER6 verification effort as a means of recreating postsilicon found bugs via simulation. This method, which was referred to as the *irritator thread* method, generated one pass of an infinite loop containing a short instruction sequence on one or more threads. These infinite loops ran as irritator threads, whereas one thread, referred to as the *primary thread*, executed the standard-generation-type random instruction stream used on prior projects. When the primary thread finished its code, it terminated the infinite loops running on the irritator threads by storing a no-op instruction into the irritator threads' instruction streams in place of the branch. This is illustrated in **Figure 4**. This method had several accomplishments. First, since the irritator threads were very short to generate, they were very efficient, although they run as long as the primary thread runs. Second, they excelled at targeting cross-thread verification scenarios since they tightly constrained the irritator thread code to do one task very often while the primary thread ran more random code sequences. The result was a nearly guaranteed controlled cross-thread interaction, with a secondary benefit being that all the threads ran exactly the same amount of time during simulation. Although this method was created for the two-threaded POWER6 core, it proved very beneficial when extended to four threads on the POWER7 core.

Although the irritator thread method excelled in finding bugs requiring a cross-thread interaction, it did have several shortcomings. First, since the irritator threads executed in

infinite loops, there was little chance of finding a bug on those threads because of their lack of randomness. In order to achieve the randomness of the standard generation method with dramatically improved test generation performance, a second test generation method was created that generated several single-threaded test cases and then merged those tests into a single four-threaded test in many different combinations. Because of the nature of SMT processors, simply changing which tests are mixed together makes each resulting combination unique and, thus, capable of finding unique bugs.

The final method for generating new tests was similar to the merge approach. Like the merge method, single-threaded tests were generated. However, instead of mixing different single-threaded tests together, we replicated the single-threaded tests to all four threads. Since this method required only one thread to be generated, it was very efficient. In addition, it provided a means of testing certain microarchitecture features in the processor core that are exercised only when multiple threads are running the exact same instruction stream with the same address translation mappings.

Another major advancement to the verification effort for POWER7 was the core directed test plan (DTP). While POWER5 utilized a DTP, the POWER7 DTP dramatically expanded the scope of the plan, adding much more detail and better organization. To begin with, the POWER7 DTP was created using a spreadsheet format with tabs for each core unit and architecture feature. Each tab contained line items addressing every aspect of the design requiring specially targeted test cases. These line items were initially created by the verification lead engineer and were augmented after detailed unit-by-unit microarchitecture design reviews with the design and verification teams. Each item consisted of a description of how to stress that aspect of the design—that is, test developer owner, projected and actual completion dates, and relative priority and coverage tag.

The coverage tag was a new concept in POWER7 that provided a means to correlate line items in the DTP with groups of related coverage events, thus closing the loop between the DTP and the coverage reports. More on this is provided in the next section of this paper, but it is worth mentioning here to convey that the DTP was much more than just a test plan; it was the means by which the core verification effort could be measured throughout the project in terms of schedule and completeness.

The DTP enabled work beyond the traditional core verification effort. It also served as the starting point for developing targeted exerciser shifts for the simulation acceleration platforms. With both the traditional core verification team and the EoA team working from a common test plan, there was a much closer working relationship between the teams than on previous designs. This methodology change fostered much more targeted and higher quality exerciser shifts than on past designs. Since the same

coverage data could be collected from both platforms and then compared side by side, tradeoffs and analysis could be made to optimize the verification efforts.

Coverage

POWER7 adopted new methodologies and technologies related to coverage that helped to improve the quality of the verification process and provided useful information on the state of the verification process. These new methodologies and technologies can be classified into three main categories: definition of a high-quality coverage plan, coverage analysis and reports, and taking coverage to acceleration and postsilicon. As mentioned in the section on unit verification methodology, most of the coverage in POWER7 was done using *BugSpray* annotation in the RTL. While other coverage tools with a top-down and architectural view, such as Abacus and Meteor, were also used, their usage was minor.

In terms of the coverage plan, the POWER7 team spent extra effort to ensure that all aspects of the design were included in the coverage plans. This included ensuring that all the units have good coverage events defined and that coverage is measured (and looked at) everywhere. Ensuring good coverage definition was done using reviews of the coverage events defined by each unit and correlating them to the main features of the unit and its verification plan.

The DTP is a good example of the benefits resulting from a tight connection between the verification plan and coverage. The connection between the DTP and coverage was achieved by using tags that were defined for each item in the DTP and coded in the coverage events. This connection between the test plan and coverage events provided important feedback during the definition of the coverage plan. It helped identify items that do not have enough coverage (tags with no events associated to them) and missing items in the test plan (events that cannot be associated to any tag). The second benefit the DTP provided the coverage effort was the ability to focus on specific items when looking at coverage data. For this purpose, DTP coverage reports provided the state of each category and tag in the test plan to provide a broad picture of the coverage status and helped identify weak areas.

The DTP coverage report is one of several reports and analysis mechanisms developed for POWER7. The goal of these reports is to provide important information to the users without overwhelming them with too much data. In addition, the reports also provided the ability to navigate in the coverage space and dive deep into the coverage of specific events. This was done by postprocessing the coverage data and presenting it in easy-to-read Hypertext Markup Language (HTML) reports. The HTML reports provided simple mechanisms to highlight important information (such as uncovered and lightly covered events) and navigate between reports and levels of detail using

hyperlinks. To allow easy comparison between coverage from various sources, the reports show the coverage at various levels in the design hierarchy or by various generation tools side by side. The reports incorporated other features to better highlight important information and ignore irrelevant data. These included filtering of specific events, highlighting impossible-to-hit events and aged-out events (i.e., events that have been hit in the past but not recently hit).

An additional type of coverage analysis was provided by the *BugSpray over time* (BoT). Contrary to the HTML reports that tried to minimize the amount of data shown to the user, BoT keeps a lot of information in its database, specifically the coverage of each event in each model by each test template in each day. With this information, BoT can, for example, grade test templates based on their coverage performance and find test templates that hit hard-to-hit events. The combination of the HTML reports and BoT allows the users to look both at the big picture of coverage and dive into the details when needed. This, in turn, improves the benefits of coverage to the project and allows targeted testing based on coverage if needed.

The third aspect of coverage-related enhancements done in POWER7 is the increased importance of coverage in postsilicon validation and the postsilicon exercisers. EoA was one of the main beneficiaries of the DTP and coverage analysis and reports introduced in POWER7. While Awan and exercisers have been used in the past, POWER7 is the first project where exerciser development was extensively guided by BugSpray coverage from Awan. The DTP provided input to the exercisers teams, which enabled earlier exerciser high-level design and development. The coverage reports allowed the logic owners and the exerciser owners to identify holes and improve the quality of the exercisers and readiness for bring-up.

The POWER7 core DTP and coverage information collected in EoA were some of the sources for the postsilicon bring-up test plan. In general, the test plan comprises a set of items that are considered important to be exercised during bring-up. For each such item, the bring-up team identified a set of relevant coverage events on the one hand and relevant exercisers and test templates on the other hand. Detailed analysis of EoA coverage data (from BoT and other sources) helped ensure that the relevant test templates did hit the target coverage events of each item in the test plan. When this was not the case, the analysis found exercisers and test templates that hit the events, giving priority to test templates that are more related to the item, and finding a minimal set of test templates. If the coverage events were not hit at all, the coverage analysis reported this to the bring-up team and corrective measures (such as adding new test templates) were taken.

Exercisers on accelerators

For POWER7 verification, IBM's long lineage of cycle-based hardware accelerators of the *AWAN* family [3] was extended with a third-generation machine called *AwanStar*. IBM continues to develop these hardware accelerators to support the high model capacity requirements of its high-end server multichip system model simulation needs. The new generation is able to provide a simulation speedup of up to a factor of 20 times over the previous model (*AwanNG*). All accelerator generations share a seamless model build flow with each other as well as with Mesa. This allowed the POWER7 team to choose the most appropriate simulation engine for a given task from a range of several orders of magnitude in speed.

Verification using the AWAN family of hard accelerators initially began as an activity for the hardware bring-up team in preparation for running the real hardware on the test floor. This originally consisted of pervasive verification only. Hardware exerciser code was typically run a few weeks before tape-out or between tape-out and the arrival of parts, as a check that the pervasive simulation put the hardware into a valid run state. These laboratory exercisers can be thought of as self-generating self-checking test code. For POWER7, an effort was made to aggressively use the Awan resources as an extension of mainline simulation, in an attempt to find bugs and to prepare for hardware bring-up. This effort is called EoA.

The EoA environment is a derivative of the mainline simulation environment. It moved accelerator simulation a step away from the pervasive and laboratory world. A key enabling factor to allow verification work to shift and choose between the different engine platforms was the use of synthesizable specifications in key areas. Any synthesizable specification can be directly mapped into a cycle-based model of any of the verification engines. A good example of this strategy was the use of an L2 behavioral written in synthesizable VHDL, allowing it to be reused as a component in core standalone simulation on any of the simulation engines, including the Awan accelerators. EoA uses the same models. It also uses the same C++ code for initialization and the infrastructure to autosubmit jobs, count cycles, count passing test cases, and detect failures, as well as to tabulate coverage.

Another key factor in the "shift-left" approach used in POWER7 was a tighter linkage of presilicon and postsilicon verification, much of which was centered on the development of the postsilicon exerciser tools. Running postsilicon exercisers on the hardware accelerators provided additional and complementary verification stimulus to the presilicon effort. Utilizing presilicon coverage tracking while running postsilicon exercisers allowed the postsilicon team to prepare and target their exerciser code much earlier than ever before.

Laboratory exercisers are biased to generate random tests from a group of instructions or to generate specific

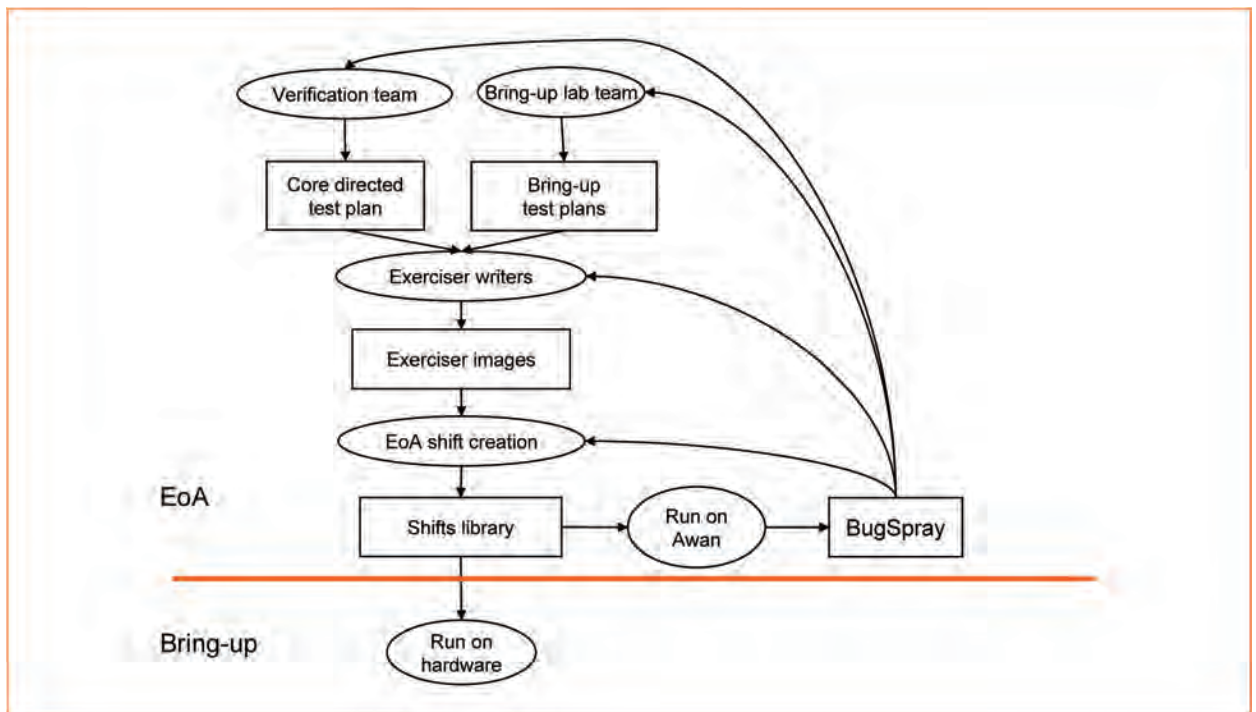


Figure 5

EoA exerciser workflow.

instruction sequences. When combined with randomization techniques, this is referred to as an “exerciser shift.” For POWER7, EoA aligned its exerciser test plan with the core DTP developed and used by mainline core simulation. This meant that the exerciser shifts run in EoA would be tracked to the same line items as mainline simulation. To prove whether EoA was covering these items, coverage results were collected and analyzed for every shift. Coverage was used by the exerciser developers to see what was not being hit and to drive improvements to the exerciser code. On previous projects, the exerciser developers could only make statements about what they believed the exercisers were running. On POWER7, well-conceived coverage events provided proof.

The POWER7 exerciser workflow is shown in **Figure 5**. There are two initial sources of requirements for the shifts: the DTPs from the verification team and the laboratory test plans from the bring-up laboratory team. Starting with these, the exerciser team makes initial exerciser biases “open loop,” based on the test plans, the architecture, and the microarchitecture. The EoA team combines the exerciser images with chip and system models of various configurations to create the “exerciser shifts” run on Awan. The coverage loop is closed with BugSpray coverage: Each shift iterates until it hits the right BugSpray events, and the entire suite of shifts is evaluated against the mass

of the BugSpray events. When initial hardware arrives, a shift “library” has been built up that serves as the initial set of test cases on hardware.

EoA has several advantages over mainline simulation that help find a different class of problems. First, EoA is superior in raw performance. For POWER7, simulation on Awan achieved up to 175,000 cycles/s, depending on the size of the model. Mainline simulation is multiple magnitudes slower. Microarchitected functions that take thousands of cycles of simulation time can be tested more often and in various random scenarios using acceleration. Functions that fit into this category for POWER7 are recovery, error injection, and power management operations.

Second, EoA did not have dependence on traditional verification checkers. This freed testing from the development overhead and debug of these software checkers and allowed for function to be tested if the checkers were not ready. This allowed for quick feedback on new or rarely used modes and workarounds. To compensate for the lack of software checkers, EoA relied on checkers written in VHDL, hang detection, and miss-compares detected by the exercisers.

Finally, it provided a means to boot Linux** and other operating systems. This allowed for performance analysis and the execution of customer workloads consisting of millions of instructions prior to tape-out.

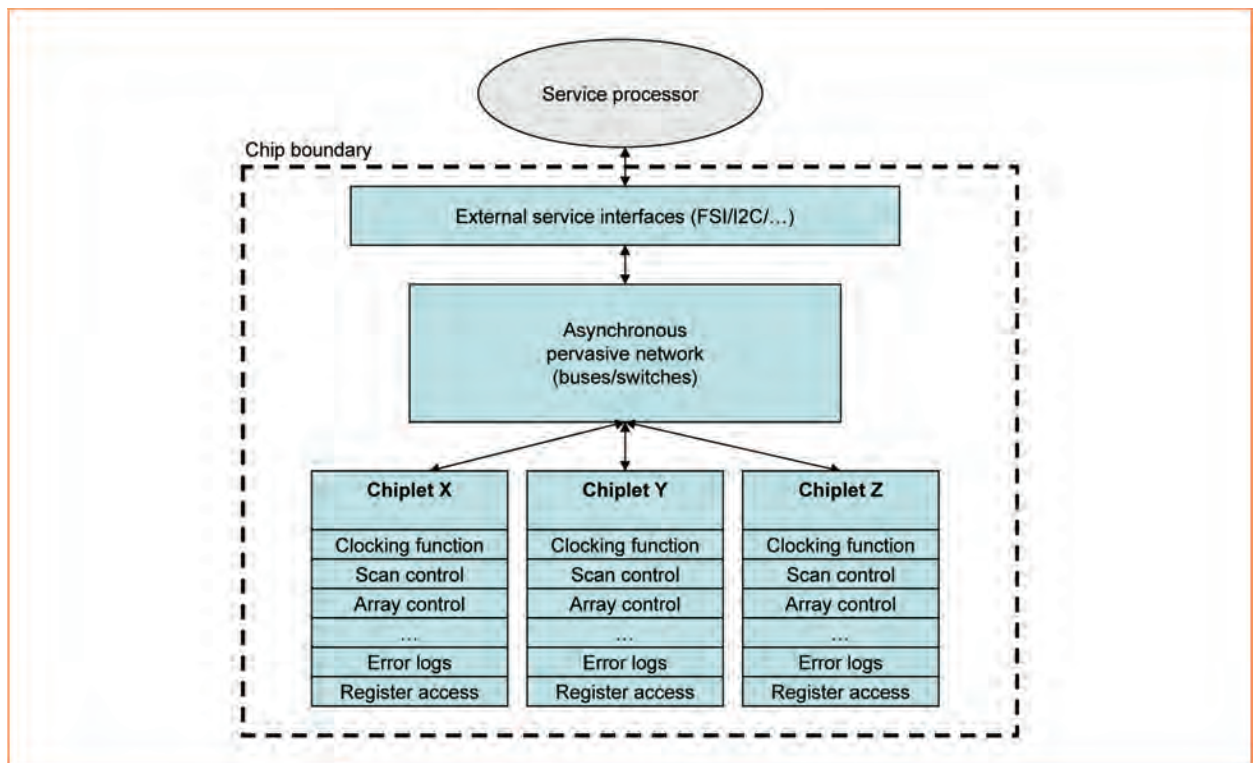


Figure 6

IBM POWER7 pervasive design structure.

The next step for EoA was to translate the simulation shifts to the hardware bring-up environment. The core DTP provided the basis for the laboratory test plan. The library of exerciser shifts, more than 2000, was available to run in the laboratory with known capabilities as to what the shift covered. The laboratory team was able to choose a subset of the shifts that had the highest value based on examining the coverage reports on a shift-by-shift basis.

For POWER7, EoA was successful in delivering its goals of finding bugs and preparing the exerciser shifts for hardware bring-up. Compared with POWER6, EoA delivered an increase of more than 3× in the number of bugs found. A significant portion of these would have escaped to the hardware. The POWER7 bring-up was very successful. The laboratory team spent little time developing exercisers and were able to focus on debugging hardware problems from the first power-on test. The exerciser coverage shielded firmware and operating system developers from finding hardware bugs.

Pervasive function verification

The pervasive design of a chip includes all the logic that initializes the chip and also provides the service interface for all RAS features. Proper scan function, clock setup and

distribution throughout the chip, register access, and array access and initialization all play a role in this and are considered pervasive functions. Historically, it also includes logic to detect and react to problems within the chip, and built-in self-test (BIST) functionality to detect and repair problems in the arrays and latches.

On POWER7, the pervasive design has grown to include many new complex features and functions (Figure 6). This includes logic that allows direct memory accesses from a service processor, as well as function to give mainline nest units access to registers within any chip in the system. In addition, increased temperature and workload sensor capabilities were added to the design, along with significant features in the area of dynamic power management and gating functionality.

One of the most dramatic design changes on POWER7 was the new hierarchical organization of the chip's pervasive units. As illustrated in Figure 6, the chip is split into *chiplets*, and each contains common instances of various pervasive logic macros such that the pervasive function for a given chiplet (i.e., clock grid, power plane, scan rings, and array initialization) is completely independent of the equivalent function on any other chiplet. The pervasive macros within each chiplet are accessible to external service interfaces

via an asynchronous network of cascaded buses, switches, and slaves.

Structural verification

With increasing demands from technology scaling, power management, and RAS and pervasive design requirements, the amount of topology-based or structural design rules has been growing significantly. This created new challenges for pervasive verification and required a paradigm shift in the verification strategy. The sheer volume of clock-control-related connections within chiplets and the high complexity of the rules associated with these controls called for a structural analysis approach. The main challenges included the following:

- Conversion of technology and test requirements into a complete set of crisp logic design guidelines that can be checked against an RTL netlist.
- Management of the large number of rules, as well as the necessary exceptions and waivers for special cases. This had to be done in an adequate fashion under the constraint that any solution has to work for the inherent design hierarchy with multiple component instantiations.
- Balancing the effort to enforce the rules on the design side and therefore restrict the possibility to create innovative solutions in some cases versus the verification effort in maintaining additional rules, exceptions, and waivers.

The most effective way to check such structural rules is by static checking, that is, topology checks on the RTL netlist level, rather than waste simulation cycles to find downstream functional bugs caused by wrong structural connections. The plethora of already existing and ever-increasing rules creates a significant maintenance problem if the verification team has to write custom code for all these static checks. For this reason, the POWER7 pervasive verification team developed a new easily configurable structural checking tool called *Spider* [14]. Spider uses a tree-grammar specification of structural design rules and employs an open-source parser generator tool to generate the checking code for the rules. Initially targeted to verify proper clock connectivity and adherence to technology rules of the POWER7 design, this approach led to a highly productive and easily adaptable tool that was broadly used across the POWER7 project.

Asynchronous verification

A 45-nm node chip as large as POWER7 with its power management capabilities requires the use of multiple phase-locked loops and asynchronous clock domains at an unprecedented level for a high-end processor chip. This makes efficient asynchronous clock-domain verification a key challenge. Our approach was twofold. We used well-known static topology-based cross-domain crossing

(CDC) verification, which identifies and checks CDC interfaces for design rule compliance. In addition, IBM's CDC tool is also able to insert intelligent virtual skewing logic into the CDC paths, which inserts biased-random delays into the signal flow. This flow is completely integrated into the cycle-based RTL simulation flow and allows stressing of CDC boundaries at all levels of verification.

Certain pervasive functions such as logic BIST (LBIST) are also extremely sensitive to nondeterministic data sources. First used on POWER6, the use of IBM's CDC tool uncovered numerous isolation bugs and helped identify the respective design components. During POWER7 design, the complexity of the asynchronous design drove additional development to improve the controllability of the tool and became an integral part of the regression suites.

Enhanced from the POWER6 methodology was the use of machine-readable formal specification for different design aspects. We heavily utilized parsed formal descriptions for the verification of array ABIST and array properties, as well as for register specifications. One recurring problem for these formal approaches was the management of exceptions and incomplete specifications. The specifications tend to be incomplete for reused designs. This will be one area for future development in pervasive verification techniques.

Unit-level pervasive function verification

Aside from these structural and formal techniques, the pervasive verification team introduced a full suite of Fusion unit environments utilizing constrained random techniques, as described in the section on unit verification above. Traditionally, verification of the pervasive logic of a chip has focused on the bring-up aspect of the design. The majority of simulation cycles devoted to testing this area were run in the form of procedural tests executed in similar order to the power-on sequences run in the laboratory. Most checking in this type of environment was of the end-to-end variety with results being checked between command accesses instead of on every clock cycle.

Given the increased complexity in the POWER7 pervasive design and the addition of mainline functional behavior in many pervasive macros, adding a unit-level verification approach became necessary. Similar to the processor functional unit environments described earlier, Fusion C++ code was developed to drive random stimulus into individual pervasive components. Accurate cycle-by-cycle checking ensured proper behavior of unit outputs and observable registers.

The ongoing theme in both the design and the verification team for POWER7 was commonality and reuse. For the pervasive design, this translated into packet based networks to deliver pervasive control and data across the chip as opposed to dedicated point-to-point connections. As a result, most pervasive components shared a common access interface that

allowed for common Fusion interface and driver code to be reused between several pervasive unit environments.

Some components in the design were instantiated in multiple places within the chip but slightly differing in behavior. To avoid making altered copies of the same component for every version within the chip, the design team often made use of VHDL generics so that only one version of the component could be repurposed to support the variety of behaviors required. Similarly, the effort was made in verification to support one version of a given unit checker, tailoring the differences in expected behavior through input biases or, in some cases, C++ class inheritance. Such techniques were also necessary to allow the Fusion unit environments to be reused across the multitude of projects for which the pervasive design remained common.

The unit environments proved to be extremely successful uncovering a high number of design defects across all areas of the pervasive design. As with mainline functional unit verification, there was also a need to run tests at a higher level of the design hierarchy to uncover problems dealing with multiunit interactions outside of a single unit environment's scope. While the pervasive bring-up and end-to-end environment could qualify as such a higher level environment, it was somewhat limited in terms of checking capability and test-case configurability. Instead, we merged all the pervasive Fusion unit environments into a single model, which is referred to as the *pervasive element environment*. It is composed of unit monitors, IMs, and a subset of the drivers.

An additional Fusion driver specifically for the pervasive element environment was developed to generate code sequences with some randomness in timing and data to target scenarios that promote a multiunit interaction within the pervasive design. This driver initiates commands into the logic at the top of the pervasive network. This allows natural interaction and "handshaking" between real design components from the insertion point all the way down to the various asynchronous endpoints within each chiplet. Since all pervasive Fusion unit monitoring code is included in this environment, all activities within every unit are fully checked on a cycle-by-cycle basis within each pervasive element test. Eventually, the pervasive element environment became the vehicle of choice for laboratory support because it marries the flexibility of test control from Fusion with the large design boundary scope that was once only available in the end-to-end environment.

Power management verification

POWER7 provides an unprecedented level of power management capability for IBM server processors. Features added for POWER7 include new architected idle states, wide-range dynamic frequency scaling per core, chip- and package-level automated voltage control, dedicated power and performance monitoring hardware, and several features

Table 1 Power management timescales.

Capability	Timescale
Frequency slewing	10–50 μ s
Voltage slewing	100 μ s–1 ms+
TPMD communication	250 μ s
Sleep exit	1 ms
Sleep entry	1 ms+
TPMD control	1–32 ms
Hang timers	8–300 ms

for autonomous power management. A full exposition of these capabilities appears in [1].

Many of the power management features were covered using the unit- and core-level techniques discussed previously. Here, we describe the new core-, chip-, and package-level power management protocol verification required for POWER7. The environment described below is derived from the EoA methodology discussed earlier.

Power management protocols operate between units in the processor cores, over the pervasive infrastructure inside the chip, as well as over the coherent interconnect inside the multichip modules that comprise high-end servers. Inputs to the power management system include the core executing idle instructions, the system environment providing interrupts that may take the system out of an idle state, and control actions by the thermal and power management device (TPMD)—an external microcontroller present in all POWER7 systems tasked with implementing power management policies.

Power management protocol verification requires system-level end-to-end verification strategies. As examples of the protocols involved: 1) a frequency change initiated by TPMD first targets a centralized power management controller (PMC), which relays the frequency change over the pervasive network to running cores but holds the change for sleeping cores until they reawaken; and 2) individual core Sleep and Wakeup requests are coordinated through the PMC; the last core on a POWER7 to enter Sleep triggers an automatic voltage change to a minimum data-retention level, whereas the first to Wake must reestablish the operating voltage.

Time compression

From a simulation perspective, power management protocols play out over significant timescales; the execution of a Sleep instruction on a core may result in a package-level voltage change several milliseconds later. **Table 1** details timescale ranges for power management actions. For reference, the AwanNG environments typically used for power management verification simulate 2 ms/h of "wall" (actual elapsed) time. One of the major verification challenges was to develop ways to compress protocol timescales without missing bugs. Although several

time-compression techniques were developed, a subset of tests was always run with full timing fidelity.

Transparent fine-grained wide-range frequency slewing is a major advantage of POWER7. The simulation environment includes a programmable clock model that uses pulsewidth modulation to precisely model variable frequencies in the cycle-simulation environment, providing coverage of issues related to assumptions on frequency differences or timing. We typically do not model the actual slew times to speed simulation. We also typically program the on-chip voltage sequencers to operate much faster than physical constraints would allow, being careful to ensure that we leave room to observe “windows” where various autonomous processes might incorrectly produce illegal frequency and voltage settings.

Core-level idle states are only entered once every thread has executed a power saving instruction, and the POWER7 Sleep protocol requires all coherent data to be purged from the core and caches before stopping clocks and reducing voltage to a static data-retention level. These requirements are major impediments to efficient protocol verification in randomized environments due to the required synchronization and long purge latencies. Instead, we run simple easily controllable idle instruction test programs on the cores that maintain no coherent data. This allows us to safely “fake” the coherent purge in many cases. Full correctness coverage of the coherent purge is provided by the EoA environment discussed earlier.

The external TPMD microcontroller communicates with POWER7 over a relatively slow I²C (Inter-Integrated Circuit) interface, using the programmable on-chip communications assist (OCA) unit to efficiently collect temperature and performance data over the pervasive network. In simulation, we use OCA to irritate the power management infrastructure up to 1,000 times faster than possible with a TPMD model, stressing the design and design assumptions. Finally, state machine hangs were the most commonly observed symptoms of protocol problems. Using the time-compression techniques discussed above, we are able to reduce hang recognition times to 100,000–1,000,000 cycles, from multimillions of cycles in the full EoA environment.

Simulation environment

Our end-to-end test scenarios typically move the system from one power management state to another using a directed random sequence. Efficient simulation requires fast control of the core-level test programs and immediate recognition of power management steady states; there is little need to burn cycles simulating running cores or sleeping chips. Because of the possibility of short-lived voltage or frequency hazards caused by interactions of autonomous mechanisms, every system-level voltage and core-level frequency change needs to be immediately checked for correctness. Thus, power

management verification requires a more significant level of interaction between the test drivers and checkers than has typically been the case in hardware simulation accelerator environments.

Although the low-level hardware accelerator application programming interfaces provide a very efficient interaction with the model, the Fusion software stack used for unit-level and other system-level verification is not optimized for this type of run–stop–check–run use when only a few dozen cycles may transpire between simulation stops. This led to the development of a dynamic-language “thin client” for the hardware accelerator that provides barebones abstractions and short code paths between test programs and the simulator. The well-developed breakpoint facility is used to observe and control tests by recognizing interesting machine states and efficiently directing stimulus (e.g., by only delivering interrupts to processors around the time of their idle state transitions).

It is a testament to the skill of our simulation hardware and software development teams that we are able to maintain 80% of the theoretical performance of the hardware accelerator despite interrupting simulation every 200 cycles on average. The full EoA environment typically simulates six system idle state transitions per hour. Using the time-compression techniques discussed earlier, we achieve more than 300 idle state transitions per hour of simulation in the power management environment. This throughput is critical; one single-cycle protocol error window, later computed to randomly occur approximately once per 25,000 Sleep sequences, was observed *exactly* once in hundreds of hours of hardware-accelerated simulation.

Results

Power management is one example that demonstrates the success of the POWER7 verification activities. Despite a late verification start in this area, the first-silicon chips produced worked remarkably well, providing all features required for manufacturing, characterization and development of power management firmware, Hypervisor, and AIX* code. The verification and laboratory experiences led to enhanced power management coverage in our preexisting exerciser programs, as well as to a new class of exercisers specifically for power-related testing.

Overall, the results from POWER7 clearly show that innovation in multiple key areas coupled with a very structured process achieved excellent hardware quality already for the first tape-out. This success is based on a very robust unit verification methodology driven by an experienced team, which found almost 60% of all the defects (**Figure 7**). Even for the memory controller, which posed a struggle throughout the project, the results were satisfactory. In this paper, we have described the enhancements in core and chip pervasive verification. Those

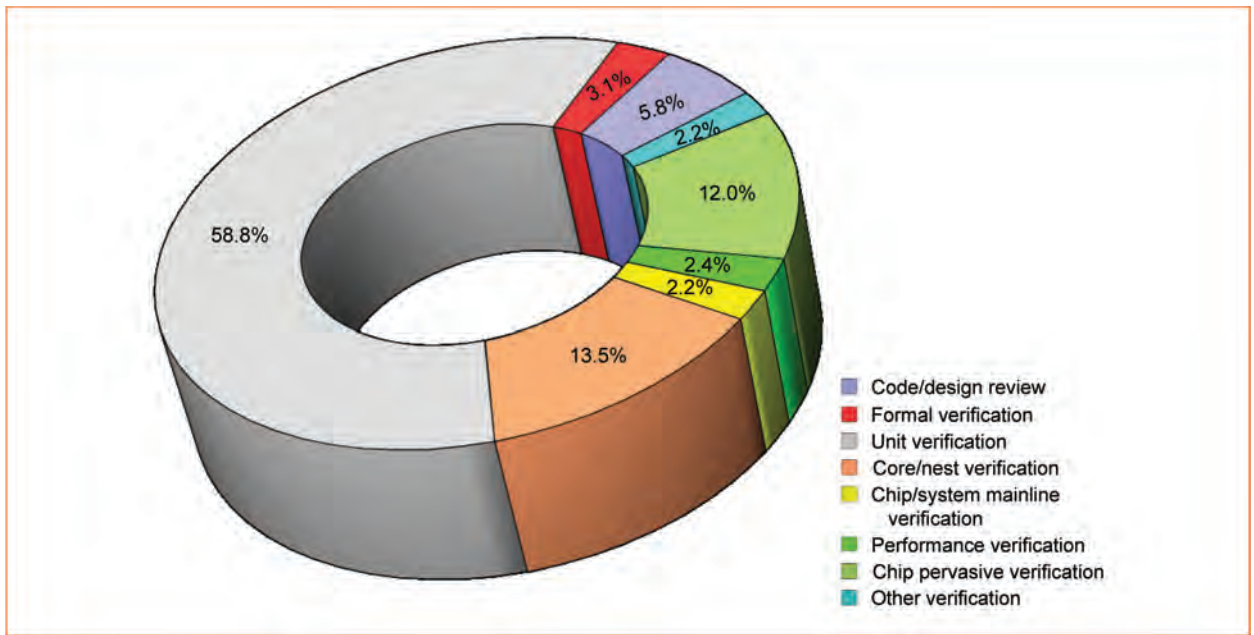


Figure 7

Defects found in functional verification by activity.

verification activities fueled by improved methods are accounting for finding another 25% of the defects.

The remaining defects have been found in different environments. However, their relative share does not accurately reflect the importance of these environments. FV, for instance, accounts for 3% of all the defects, which translates to a few hundred fails; however, the figure does not reflect the much bigger share of problems that were found and fixed by the design team before the design was ever given to the verification team. Furthermore, many of the fails found by FV would have escaped all the other environments. Similarly, chip and system mainline verification found on the order of 2% of the problems, but as explained above, it also helped improve our bring-up exercises. At the same time, the small percentage underlines the quality of the unit and core-nest environments.

In the end, the quality of any verification effort is determined by the progress the bring-up makes with the first silicon. For POWER7, the feedback was outstanding. On the first day the chip was powered up in the laboratory, all types of instruction sequences could be executed out of the large on-chip cache. A few days later, the operating system was booted, and the focus shifted to verify the RAS and power management features of the design. As expected, the bring-up found problems in the range of 1%–2% of the number of defects found in verification, but none of those defects have been a “showstopper” or required heroic workarounds or

function disables that would leave large parts of the design untested.

In the future, projects will benefit from all the improvements we made across the board. In addition, we will continue to enhance our tools and methods, strengthen the link between presilicon and postsilicon testing, and further optimize our processes to deal with higher complexity and shorter schedules.

*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 Linus Torvalds, Inc., The Open Group, Sony Computer Entertainment, Inc., or Microsoft Corporation in the United States, other countries, or both.

References

1. B. Sinharoy, R. Kalla, W. J. Starke, H. Q. Le, R. Cargnoni, J. A. Van Norstrand, B. J. Ronchetti, J. Stuecheli, J. Leenstra, G. L. Guthrie, D. Q. Nguyen, B. Blamer, C. F. Marino, E. Retter, and P. Williams, “IBM POWER7 multicore server processor,” *IBM J. Res. & Dev.*, vol. 55, no. 3, Paper 1, pp. 1:1 1:29, 2011.
2. M. Floyd, M. Ware, K. Rajamani, T. Gloekler, B. Brock, P. Bose, A. Buyuktosunoglu, J. C. Rubio, B. Schubert, B. Spruth, J. A. Tierno, and L. Pesantez, “Adaptive energy management features of the IBM POWER7 chip,” *IBM J. Res. & Dev.*, vol. 55, no. 3, Paper 8, pp. 1:1 1:18, 2011.
3. J. M. Ludden, W. Roesner, G. M. Heiling, J. R. Reysa, J. R. Jackson, B.-L. Chu, M. L. Behm, J. R. Baumgartner, R. D. Peterson, J. Abdulhaziz, W. E. Bucy, J. H. Klaus, D. J. Klema, T. N. Le, F. D. Lewis, P. E. Milling,

- L. A. McConville, B. S. Nelson, V. Paruthi, T. W. Pouraz, A. D. Romonosky, J. Stuecheli, K. D. Thompson, D. W. Victor, and B. Wile, "Functional verification of the POWER4 microprocessor and POWER4 multiprocessor systems," *IBM J. Res. & Dev.*, vol. 46, no. 1, pp. 53–76, Jan. 2002.
4. D. W. Victor, J. M. Ludden, R. D. Peterson, B. S. Nelson, W. K. Sharp, J. K. Hsu, B.-L. Chu, M. L. Behm, R. M. Gott, A. D. Romonosky, and S. R. Farago, "Functional verification of the POWER5 microprocessor and POWER5 multiprocessor systems," *IBM J. Res. & Dev.*, vol. 49, no. 4/5, pp. 541–553, Jul. 2005.
5. C. A. Krygowski, D. G. Bair, R. M. Gott, M. H. Decker, A. V. Giri, C. Habermann, M. Heizmann, S. Letz, W. J. Lewis, S. M. Licker, H. Mallar, E. C. McCain, W. Roesner, N. Siddique, A. E. Seigler, B. W. Thompto, K. Weber, and R. Winkelmann, "Functional verification of the IBM system z10 processor chipset," *IBM J. Res. & Dev.*, vol. 53, no. 1, Paper 3, pp. 3:1–3:11, Jan. 2009.
6. H. Mony, J. Baumgartner, V. Paruthi, R. Kanzelman, and A. Kuehlmann, "Scalable automated verification via expert-system guided transformations," in *Proc. Conf. Formal Methods Comput.-Aided Des.*, 2004, pp. 159–173.
7. S. Ben-David, C. Eisner, D. Geist, and Y. Wolfsthal, "Model checking at IBM," *Formal Verification Test. Technol. Syst. Des.*, vol. 22, no. 2, pp. 101–108, 2003.
8. *IEEE Standard VHDL Language Reference Manual*, IEEE Std. 1076-2008, 2009.
9. R. Gott, J. Baumgartner, P. Roessler, and S. Joe, "Functional formal verification on designs of pSeries microprocessors and communication subsystems," *IBM J. Res. & Dev.*, vol. 49, no. 4/5, pp. 565–580, Jul. 2005.
10. K. Kailas, V. Paruthi, and B. Monwai, "Formal verification of correctness and performance of random priority-based arbiters," in *Proc. Formal Methods Comput.-Aided Des.*, Austin, TX, Nov. 2009, pp. 101–107.
11. C. Jacobi, K. Weber, V. Paruthi, and J. Baumgartner, "Automatic verification of fused-multiply-add FPU's," in *Proc. Des. Autom. Test Eur.*, 2005, vol. 2, pp. 1298–1303.
12. T. Gloekler, J. Baumgartner, D. Shanmugam, R. Seigler, G. V. Huben, H. Mony, P. Roessler, and B. Ramanandray, "Enabling large-scale pervasive logic verification through multi-algorithmic formal reasoning," in *Proc. Formal Methods Comput.-Aided Des.*, San Jose, CA, Nov. 2006, pp. 3–10.
13. J. Baumgartner, H. Mony, V. Paruthi, R. Kanzelman, and G. Janssen, "Scalable sequential equivalence checking across arbitrary design transformations," in *Proc. Int. Conf. Comput. Des.*, San Jose, CA, Oct. 2006, pp. 259–266.
14. C. Jaeschke, C. Schmitt, U. Herter, T. Wich, and J. Rust, "Strukturelle Verifikation mittels parser-gesteuerter Netzlisten-Traversierung," in *Proc. 13th Workshop Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen*, Dresden, Germany, Feb. 22–24, 2010.

Received March 3, 2010; accepted for publication June 25, 2010

Klaus-Dieter Schubert *IBM Deutschland Research and Development GmbH, Schoenaicherstrasse 220, Boeblingen 71032, Germany (kdschube@de.ibm.com)*. Mr. Schubert received the Dipl.-Ing. degree in electrical engineering in 1990 from Stuttgart University (Germany). Subsequently, he joined IBM in Boeblingen and has been responsible for hardware verification of various zSeries* systems and its components. He is an IBM Distinguished Engineer and the technical leader for the hardware verification of POWER* processors. He was the technical lead for the hardware verification of the z900* 2064 system before he moved to the field of hardware and software co-verification where he established the concept of virtual power-on (VPO) for zSeries and pSeries* systems. From 2006 to 2008, Mr. Schubert was on a work assignment in Austin, Texas, where he has led the verification team for the POWER7 microprocessor. He has received two IBM Outstanding Achievement Awards for his contributions in the field of hardware verification.

Wolfgang Roesner *IBM Systems and Technology Group, Austin, TX 78758 USA (wolfgang@us.ibm.com)*. Dr. Roesner received Dipl.-Ing. and Dr. Ing. degrees in electrical engineering in 1980 and 1983, respectively, from University of Kaiserslautern, Germany. He is an IBM Distinguished Engineer and verification leader for the IBM Systems and Technology Group. He joined the Electronic Design Automation (EDA) team at IBM in Boeblingen in 1984 and moved to Austin, Texas, in 1989 to join the POWER processor and systems teams. He has been technical lead for IBM server verification methodology and tools since 1996. He led the verification effort on several chip and system projects and was verification methodology lead on POWER7. He has published numerous papers, coauthored a textbook on verification, and has been invited speaker at several technical conferences.

John M. Ludden *IBM Systems and Technology Group, Essex Junction, VT 05452 USA (ludden@us.ibm.com)*. Mr. Ludden received a B.S. degree in electrical engineering from the Rochester Institute of Technology. He joined IBM in 1990 and is a Senior Technical Staff Member. His work at IBM has spanned mainframe systems, x86 processors, and UNIX** servers. Mr. Ludden served as the architecture verification lead engineer for the POWER3*, POWER4*, POWER5, POWER6*, and POWER7 processors. He has worked with the IBM Haifa Research Laboratory in Israel for 17 years to advance the primary instruction-level test generation tools used to verify all current processors within IBM. He holds several patents, has published multiple papers in the field of processor verification, and pioneered the simultaneous multithreading verification methodologies currently employed by IBM.

Jonathan Jackson *IBM Systems and Technology Group, Austin, TX 78758 USA (jrj1@us.ibm.com)*. Mr. Jackson received a B.S. degree in computer science and electrical engineering from Vanderbilt University in 1998 and an M.S. degree in electrical and computer engineering from Carnegie Mellon University in 2000. He joined IBM in 2000 and is currently an Advisory Engineer with IBM. In his time with IBM, Mr. Jackson has focused on directed random simulation for unit verification, in particular cache and memory subsystems. For the POWER7 project, he helped design the simulation guidelines and methodology and worked as part of the team that verified the memory controller.

Jacob Buchert *IBM Systems and Technology Group, Austin, TX 78758 USA (jcbucher@us.ibm.com)*. Mr. Buchert received a B.S. degree in electrical engineering from the University of Texas. He joined IBM in 2001 and is currently an Advisory Engineer with IBM. He has worked on mainline memory subsystem verification. For the POWER7 project, he was the memory controller verification unit lead.

Vireesh Paruthi *IBM Systems and Technology Group, Austin, TX 78758 USA (vparuthi@us.ibm.com)*. Mr. Paruthi is Technical Leader for formal verification (FV) application. In his role, he is responsible for directing FV methodology, strategic direction, and execution on microprocessor projects. Prior to this, he assumed a central role in the research, design, and development of SixthSense, IBM's FV and semiformal verification toolset and sequential equivalence checker, and Verity, IBM's combinational equivalence checker, and contributed to defining methodologies leveraging these tools and technologies. He has filed a large number of patents and published numerous conference papers in the area of FV.

Michael Behm *IBM Systems and Technology Group, Austin, TX 78758 USA (behm@us.ibm.com).* Mr. Behm earned an associate's degree from Lincoln Tech and is currently a Senior Engineer in the IBM Systems and Technology Group in Austin, Texas. He joined IBM in 1978 and has worked on mainframe and POWER verification and test for his entire career. He has extensive experience in the simulation environment and is currently specializing in design coverage infrastructure, planning, reporting, and analysis.

Avi Ziv *IBM Research Division, Haifa Research Laboratory, Haifa 31905, Israel (aziv@il.ibm.com).* Dr. Ziv received the B.Sc. degree in computer engineering from the Technion Israel Institute of Technology in 1990 and the M.Sc. and Ph.D. degrees in electrical engineering from Stanford University in 1992 and 1995, respectively. He is a Research Staff Member in the Simulation-Based Verification Technologies department at the IBM Research Laboratory in Haifa. Since joining IBM in 1996, he has been involved in many activities related to simulation-based verification, specifically in the areas of functional coverage, stimuli generation, and coverage directed generation. He is a member of the Institute of Electrical and Electronics Engineers and a member of the Association of Computing Machinery.

John Schumann *IBM Systems and Technology Group, Austin, TX 78758 USA (johnschu@us.ibm.com).* Mr. Schumann received a B.S. degree from the University of Texas in computer engineering in 1996. He joined IBM in 1997 and is currently a Senior Engineer in the microprocessor verification area. He has broad verification and hardware bring-up experience on POWER3, POWER4, PowerPC* 970, POWER6, and POWER7, and currently specializes in simulation accelerators and hardware exercisers.

Charles Meissner *IBM Systems and Technology Group, Austin, TX 78758 USA (cmeissne@us.ibm.com).* Mr. Meissner holds a B.S.E.E. and an M.S.E.E. degree, both from the University of Virginia. He is currently an Advisory Engineer in the IBM Systems and Technology Group in Austin, Texas. He joined IBM in 2001 and has worked on hardware validation for several generations of OEM and server POWER chips. He served as the lead for the central processing unit functional validation exerciser software for the POWER7 chip.

Johannes Koesters *IBM Deutschland Research and Development GmbH, Postfach 1380, Boeblingen 71003, Germany (koesters@de.ibm.com).* Mr. Koesters holds a master's degree in physics from RWTH Aachen University in Germany. He joined IBM in 1997 and since then has worked in verification of processor designs. After 2000 he focused on the verification of service and maintenance infrastructure of IBM chips and led this effort for chips such as the Cell Broadband Engine**, the POWER6, and the POWER7, as well as that used in the Microsoft Xbox** 360 gaming system. He holds 12 patents.

James Hsu *IBM Systems and Technology Group, Austin, TX 78758 USA (hsuj@us.ibm.com).* Mr. Hsu graduated from Duke University with B.S. degrees in electrical engineering and biomedical engineering. He also earned an M.S.E.E. degree from Stanford University in 2001. He joined IBM in 1999 and is currently an Advisory Engineer in the Hardware Verification team in the IBM Server Group in Austin, Texas. At IBM, he worked in mainline nest verification during the POWER4, POWER5, and POWER6 programs before moving to pervasive verification on POWER6. Within pervasive verification, he has worked on many projects including POWER7, the Productive, Easy-to-use, Reliable Computing System (PERCS) supercomputer, z10, and z196. Currently, he is the chip pervasive verification lead for the POWER processor. He also leads the Pervasive Fusion Skill Group, which spans verification over a variety of IBM processors and support chips and is a member of the POWER7 Chip Verification and Design Patent Review Board.

Bishop Brock *IBM Systems and Technology Group, Austin, TX 78758 USA (bcbrock@us.ibm.com).* Mr. Brock holds B.S., B.A., and M.S. degrees from the University of Texas at Austin. He is currently a Senior Engineer in the IBM Systems and Technology Group in Austin, Texas, having previously served as a Research Staff Member at IBM Research in Austin. His contributions to the POWER7 project include the development of hardware verification, microarchitectural validation, and power/performance prediction methodologies for power management hardware and firmware systems.

U-Compare: A modular NLP workflow construction and evaluation system

Y. Kano
M. Miwa
K. B. Cohen
L. E. Hunter
S. Ananiadou
J. Tsujii

During the development of natural language processing (NLP) applications, developers are often required to repeatedly perform certain tasks. Among these tasks, workflow comparison and evaluation are two of the most crucial because they help to discover the nature of NLP problems, which is important from both scientific and engineering perspectives. Although these tasks can potentially be automated, developers tend to perform them manually, repeatedly writing similar pieces of code. We developed tools to largely automate these subtasks. Promoting component reuse is another way to further increase NLP development efficiency. Building on the interoperability enhancing Unstructured Information Management Architecture (UIMA) framework, we have collected a large library of interoperable resources, developed several workflow creation utilities, added a customizable comparison and evaluation system, and built visualization utilities. These tools are modularly designed to accommodate various use cases and potential reuse scenarios. By integrating all these features into our U-Compare system, we hope to increase NLP developer efficiency. Simple to use and directly runnable from a web browser, U-Compare has already found uses in a range of applications.

Introduction

Natural language processing (NLP) tasks are generally composite in nature, and therefore, developers often need to combine multiple preexisting components to build a complete NLP system. Components solve particular NLP subtasks, such as sentence splitting or part-of-speech (POS) tagging. Individually, these components are of relatively little use, and it is only through their combination into workflows that they achieve any measure of utility.

There are many systems that support the construction of such workflows, for example, Taverna [1], Galaxy [2], Kepler [3], General Architecture for Text Engineering (GATE) [4], and Unstructured Information Management Architecture (UIMA) [5]. However, compliance with these frameworks does not necessarily guarantee interoperability of components. Additionally, they generally require knowledge of programming and a significant investment of time on the part of users to construct workflows. Of these systems, GATE and UIMA have been specifically

designed for NLP and thus will be of most interest in this paper.

GATE provides many features to developers but is basically intended as an integrated development environment for programmers (i.e., Eclipse** for NLP), whereas UIMA is intended to be more of a framework. A key difference between the two is that UIMA is component and workflow oriented, whereas GATE aims at directly assisting programming tasks. Another significant difference between the two is the data-type hierarchy present in UIMA but absent in GATE. Because of these factors, we selected UIMA as the basis for the system presented in this paper.

UIMA is a generic framework widely used in the NLP community to improve the interoperability of components. It defines, among other things, a standard workflow metadata format. UIMA was originally developed by IBM [5] and is currently an open source project at Apache Software Foundation [6] of which specification is defined as an open international standard in the Organization for the Advancement of Structured Information Standards (OASIS) [7]. Many developers provide UIMA resources, some of

Digital Object Identifier: 10.1147/JRD.2011.2105691

©Copyright 2011 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied by any means or distributed royalty free without further permission by computer-based and other information-service systems. Permission to republish any other portion of this paper must be obtained from the Editor.

0018-8646/11/\$5.00 © 2011 IBM

which include BioNLP UIMA Component Repository [8], the JULIE lab UIMA component repository (JCoRe) by the Jena University Language and Information Engineering Laboratory (JULIELab) [9], Open Health NLP [10] (Mayo Clinic and IBM), and UIMA-fr consortium [11] (University of Nantes). UIMA is also commercially used by, for example, TEMIS and Thomson Reuters, in addition to its original developer, which is IBM.

Despite all it offers, the UIMA framework is only intended to provide an abstract-level formal framework for interoperability. It leaves the implementation of its concrete structure to third-party developers, with differences in concrete implementations potentially leading to incompatibilities between tools. Our belief is that the abstract level of interoperability offered by UIMA is not sufficient, and where possible, users also need to be provided with more concrete interoperability standards. More concrete standards would usually place a compliance burden on users, but we believe this burden can be largely relieved through the use of NLP automated tools.

Our automation system, U-Compare, has been designed in a modular way compliant with the UIMA framework and allows users to reuse or customize any of its specific modules independently. The U-Compare system offers far more than just an automated workflow construction tool. An analysis of workflow results, particularly comparison and evaluation, is crucial to improve NLP systems by discovering what is wrong with particular workflows. In this paper, we describe a virtual workflow architecture that calculates possible combinations of components and runs the combinations as virtual workflows within a single UIMA workflow. This architecture allows users to compare existing UIMA components without modification. In addition to the comparison statistics, we provide visualizations of comparison results for instance-level analysis.

U-Compare is an international joint initiative by the University of Tokyo, the National Centre for Text Mining, and the University of Colorado. The whole system is available from the U-Compare website [12] under an open source Lesser General Public License.

In this paper, we begin by briefly explaining the UIMA framework. We then describe the virtual workflow architecture for comparison and evaluation. Then, we describe the U-Compare platform, which is built on interoperability and includes an easy workflow creation graphical user interface (GUI) that allows users to create a workflow in a drag-and-drop manner. We conclude that U-Compare offers a range of benefits to NLP application and component developers.

Unstructured Information Management Architecture

UIMA is an open framework specified as an international standard by OASIS. Apache UIMA provides a reference

implementation as an open source project, with a pure Java** application programming interface (API) and a C++ development kit. UIMA itself is intended to be purely an interoperability framework for all types of unstructured content such as text, audio, and video. Therefore, it is not intended to provide specific tools or type system definitions to aid specific tasks. In this section, we briefly introduce UIMA, focusing on its applications to NLP.

Data structure and data type definitions

The UIMA framework uses the “standoff annotation” style. In this style, the underlying raw text of a document is generally kept unchanged during analysis, and the results of processing the text are added as annotations with references to their positions in the raw text. This contrasts with the in-line annotation style [e.g., Extensive Markup Language (XML), Hypertext Markup Language (HTML), and Standard Generalized Markup Language (SGML)], which embeds additional information (tags) within the raw text. The basic data structure in UIMA is the Common Analysis Structure (CAS), which holds both the raw text and a set of such standoff annotations. CAS can be saved to a file or transmitted in an XML metadata interchange (XMI) format.

Every annotation should have an explicitly defined data type. Data types should be defined in a type system as part of a hierarchy of other data types. Apache UIMA provides definitions for a range of built-in primitive data types, but developers usually need to define more complete type systems based on their individual needs. In the built-in Apache UIMA type system, the top-level type is referred to as *TOP*. Below this, in the type hierarchy, we find the primitive types such as *int*, *float*, *String*, *Annotation*, and *FSArray* (an array of any annotations). (Note that in this paper, we represent type names and feature names in italics.) Types can have any number of “features,” each of which holds a primitive value or a reference to another annotation instance. For example, the basic *Annotation* type has “begin” and “end” integer-valued features that associate instances with positions in the raw text.

Components and workflows

A UIMA component is the basic processing unit in UIMA. Each UIMA component receives and updates the CAS. Component metadata descriptions should specify their input and output capabilities, listing the types of annotations they take as input and the types of annotations that they may produce as output. CASs are generated by a special type of UIMA component known as a “collection reader” and are then processed by other components known as “analysis engines” (AEs). The UIMA framework provides the ability for components to be deployed either locally or remotely as simple object access protocol (SOAP) and Vinci web services. Both locally and remotely

deployed components can be freely combined in UIMA workflows.

AEs may be either primitive or aggregate. Aggregate AEs contain a list of references to “child” AEs. In the aggregate AEs, flow controllers determine how to apply these child AEs to incoming CASs. Flow controllers can be implemented for any processing order, for example, skipping, looping, and conditionals.

A UIMA workflow is called a collection processing engine (CPE). CPEs support error handling can automatically retry failed components and may terminate execution once errors exceed a specified threshold.

Virtual workflow and comparison generator

It is desirable that a UIMA comparison architecture be able to accept any valid UIMA component without the need for modification. It should perform comparisons within the UIMA framework, and it should be as simple to use as possible; ideally, everything should be automatic, except specifying which components and which outputs to compare.

Assume that we want to compare UIMA sentence detectors A and B, which are type system compatible (details of which are described later). The most obvious strategy to compare them would be to create a serial workflow that runs A, then B, and that then runs an additional component to compare their outputs. However, this strategy has a major problem: There is no way to distinguish which *Sentence* annotations were created by which sentence detector. This could be fixed by adding a new *componentID* feature to the *Sentence*-type definition in order to identify which component created each annotation. However, this would require that a common type system be used and would likely require modification to the code of the sentence detector components.

Additionally, if we had a tokenizer component C that depended on the results of sentence detection, even the addition of a *componentID* feature would not be sufficient. The workflow is now $\{A, B\} \rightarrow C$, where $\{\}$ means a comparison flow and the component C could use the output of either A or B. A significant problem is encountered if we want to compare the performance of A or B with regard to how well C is able to perform based on their output. Not only is this a problem for comparison, but it also affects the normal serial workflows. Some components create multiple types of annotations (e.g., *Sentence* and *Token*). If only the *Sentence* type was needed and a specific tokenizer component was added, we would end up with two sets of *Token* annotations. This may cause problems for components later in the workflow, as they will receive duplicate input annotations. A common cause of this problem is that many input corpora may have certain types already annotated. For example, if an input corpus already has *Sentence* annotations, but a user wants to test a sentence detector–tokenizer pair, then the tokenizer will be presented

with not only *Sentence* annotations from the sentence detector but also those already present in the input corpus.

An alternative to this first naive solution for comparison would be to expand the original workflow into separate workflows for each possible combination of components. This would avoid the problem of subsequent components in a workflow being presented with duplicate sets of annotations and would require no modification to the components. Unfortunately, this solution is neither efficient nor appropriate for the following reasons. First, although each of the components processes the same input text, dividing a workflow into individual workflows would require the creation of separate UIMA CASs for each workflow. Second, any shared initial segment of the original workflow before the comparison components would need to be run once in each created workflow, despite that they would be running on the same set of input data every time. Finally, the comparison part somehow needs to merge the results of all the created workflows together so that it can perform comparisons. This would require operations not easily possible within the UIMA framework. The first two points make such an approach highly inefficient computationally, and the final point causes problems for compatibility with the UIMA framework. Therefore, we also reject this approach.

Another possible solution would be to use the UIMA subjects of analysis (Sofa) feature, which allow the holding of multiple sets of annotations separately within a single CAS. This would allow multiple sets of annotations produced by the compared components to be held and then passed to subsequent components in the workflow as needed via a custom UIMA flow controller. Unfortunately, this solution also has several problems. First, each Sofa independently holds its document text, whereas we need to share the same document text between all components. This makes this solution slightly inefficient, particularly when dealing with large documents. Second, components would need to be modified to read and write a specific name of Sofa, with each combination of components having a unique name. UIMA offers Sofa mapping, which could be used to avoid this problem, but Sofa mapping is not available on remotely deployed AEs. Finally, support for multiple Sofas in a single CAS is only supported by some components, so if we were to adopt this solution, we would not be able to support all valid UIMA components. For these reasons, we also reject this solution.

The solution that we eventually adopted is based on a virtual workflow architecture that avoids all of the issues encountered by the previously suggested solutions. The virtual workflow architecture that we propose does not force components to use a specific type system or require existing components to be modified. This solution does require that the input/output (I/O) capabilities of the components be properly defined. The reasons for this are described as follows.

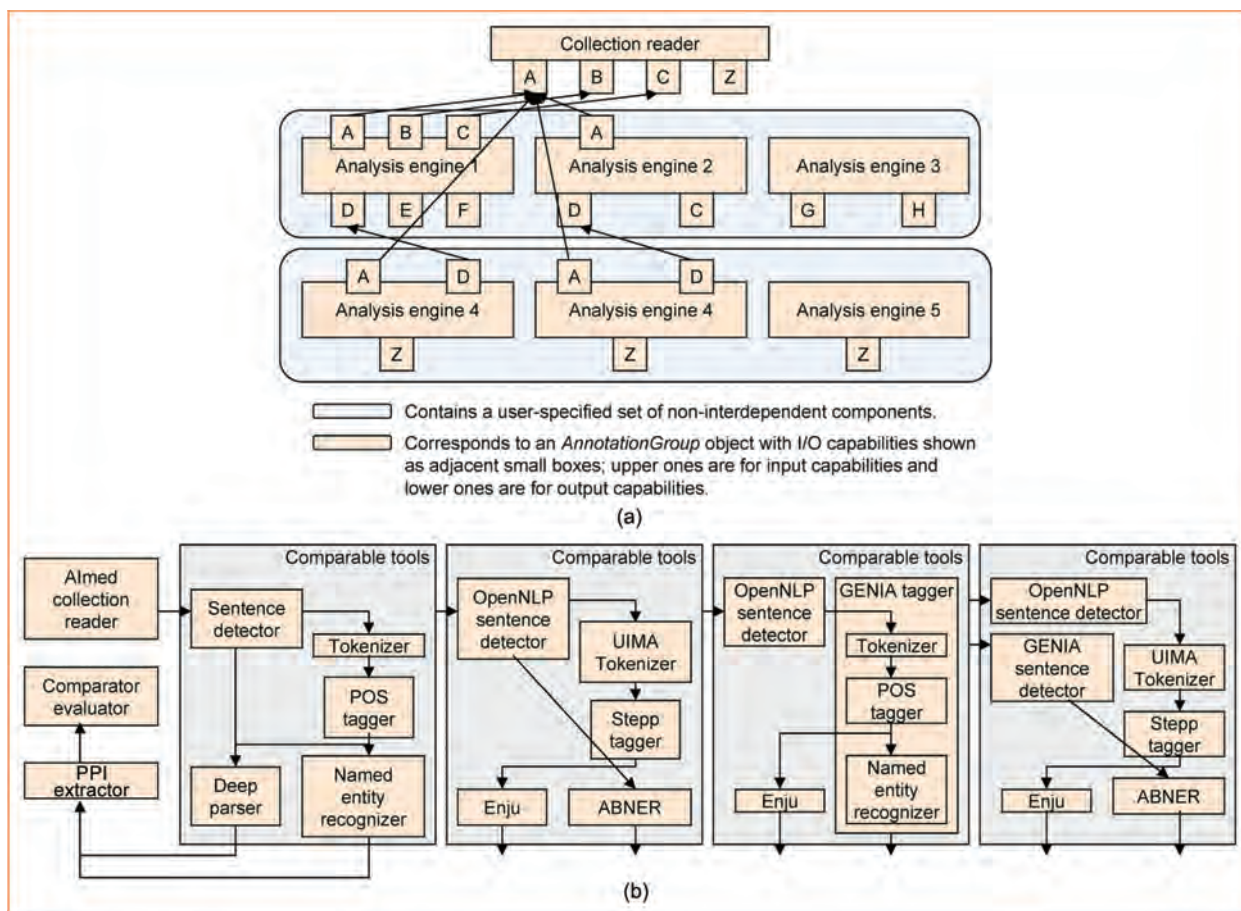


Figure 1

(a) Example of interdependent *AnnotationGroup* instances. (b) Conceptual diagrams of a PPI system workflow showing component dependencies (from left to right: general setting, basic pattern, complex tool pattern, and branch flow pattern). (CAS: Common Analysis Structure; PPI: protein-protein interaction; POS: part of speech.)

Annotation group FS

The basic mechanism of the virtual workflow architecture is that all collection reader and AE components in a workflow have one or more corresponding *AnnotationGroup* feature structures (FSs). Conceptually, the *AnnotationGroup* type is used to store the outputs of that component in order to keep them separate from the outputs of the same type generated elsewhere and to specify what inputs were used to generate them. The inputs are specified by intercomponent dependencies pointing to the *AnnotationGroup* instances from which the inputs to that component were taken. **Figure 1(a)** shows an example diagram of these intercomponent dependencies.

As shown in the figure, two or more *AnnotationGroup* instances will be created for a single AE when there are two or more possible combinations of output capabilities that satisfy the input capabilities of that AE. There is no

duplication of annotations by storing outputs in this way, as each *AnnotationGroup* corresponds to a unique set of prior *AnnotationGroup* instances.

The set of *AnnotationGroup* instances present in a CAS forms a directed acyclic graph dependence structure of intercomponent dependencies. The path of dependences from any source vertex to any terminal vertex represents a single virtual workflow. The result of a virtual workflow can be obtained by taking the union of annotations present in the *AnnotationGroup* instances along the corresponding path. It should be noted that we assume that once annotations are added to the CAS in a virtual workflow, they are not removed in other components that correspond to other vertices; this assumption is safe because removal of annotations is extremely uncommon in UIMA.

We provide a special aggregate component called a *U-Compare parallel component*. This parallel component is

used to explicitly specify a set of components that do not depend on each other and allows predefined I/O capabilities of components to be ignored when required.

Type system for comparisons and component dependence descriptions

The I/O capabilities of the UIMA components, which are represented as a list of types, are basically the only machine-readable information describing the behaviors of the UIMA components. Thus, the definition of a type system also influences what our system can deduce about component behavior. If I/O capabilities are properly defined, our system can determine which components perform comparable tasks. For example, two different components that both have *Token*-type outputs are valid targets for comparison.

Detecting components that can be compared is complex.

Figure 1(b) conceptually shows a protein–protein interaction (PPI) extraction system workflow. A PPI extraction system detects expressions describing PPIs from biomedical texts. In the example, the PPI extractor requires the outputs of a sentence detector, tokenizer, POS tagger, and a protein named entity recognizer (NER) and a syntactic parser as input. If we include two or more components for some of the required tasks in the workflow (e.g., two sentence detectors and three POS taggers), then any combination of these tools will form a virtual workflow (e.g., giving $2 \times 3 = 6$ virtual workflows). Matters are complicated by the fact that a single tool may perform multiple tasks in a single workflow [e.g., in Figure 1(b), the GENIA Tagger performs tokenization, POS tagging, and NER recognition].

Type systems should be designed with these intercomponent dependencies in mind. For example, although both a tokenizer and a POS tagger output “tokens,” they represent semantically different objects that may appear together in a single workflow. For example, in the U-Compare type system, we define both *Token* and *POSToken*, which allows us to better compare the outputs of components producing these types of annotations.

Virtual workflow processing

Using the virtual workflow architecture, a comparison can be run within a single UIMA workflow. **Figure 2** is a diagram of a comparison workflow with UIMA components shown as blocks. Assuming that components have had their I/O capabilities specified, any UIMA component can be included in such a workflow, the set of virtual workflows is automatically generated, and their results are compared. In this section, we describe how virtual workflows are processed in a step-by-step manner, and in the following section, we look at how they are compared.

The first component in a comparison workflow must be a user-specified collection reader, as is the case with all UIMA CPE workflows. The collection reader runs as it normally does.

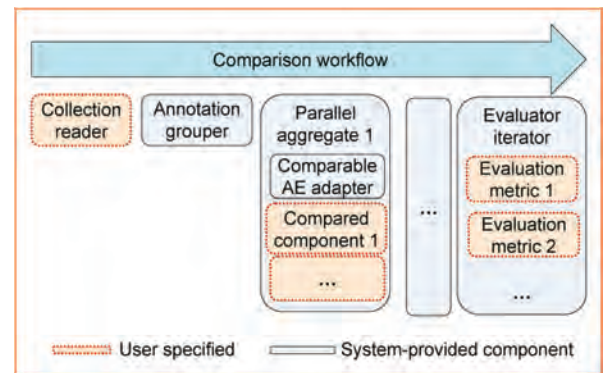


Figure 2

Block diagram of a comparison workflow. (AE: analysis engine.)

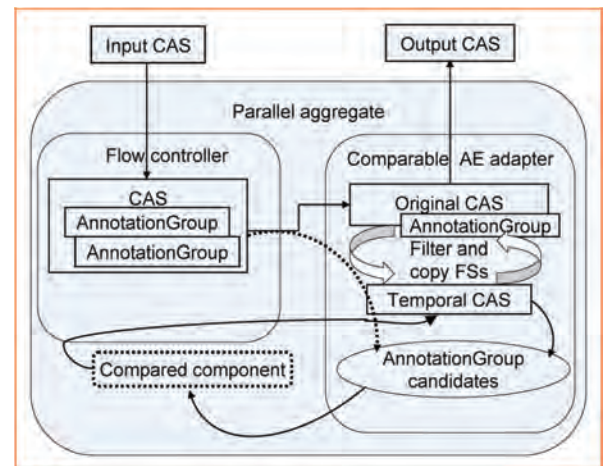


Figure 3

Block diagram of parallel component processing. (CAS: Common Analysis Structure; FSs: feature structures.)

The second component should be an annotation grouper, with output capabilities matching those of the collection reader. The role of the annotation grouper is to produce an *AnnotationGroup* instance based on the outputs of the collection reader.

Following the annotation grouper, we may have any number of parallel aggregates, each of which is an aggregate component with a special flow controller. **Figure 3** shows a block diagram of the processes inside a parallel component. The parallel component detects those previously created *AnnotationGroup* instances that satisfy the input capabilities of each of its child components and creates a new CAS containing the required inputs for that component based on the contents of the matching

AnnotationGroup. It then runs the child on this newly created CAS and copies the results of this child process back into the original CAS, grouped into a new *AnnotationGroup* instance that links back to the original *AnnotationGroup* from which the inputs were taken. This process is repeated for all possible input *AnnotationGroup* instances matching the child components of each of the parallel aggregates. We will step through it in detail in the remainder of this section.

The parallel aggregate itself is built as a standard UIMA aggregate component that has a *ComparableAggregateFlowController* as its flow controller and a *ComparableAnalysisEngineAdapter* component as a child. When the parallel aggregate component receives a CAS, it selects the first of its child components and stores its I/O capabilities as FSs in the CAS. It then passes the CAS to the *ComparableAnalysisEngineAdapter*, which will use the I/O information to determine which *AnnotationGroup* instances in the CAS are possible inputs to that component. Using CAS is the only way a parent component can communicate with a child component within the UIMA framework.

The *ComparableAnalysisEngineAdapter* then dynamically calculates which *AnnotationGroup* instances in the CAS satisfy the input capabilities of the selected child component. This calculation is based on the *AnnotationGroup* instances produced along the virtual workflow up to that *AnnotationGroup*, which can be obtained tracing ancestors.

Here, “type A satisfies type B” means that the type A is equal to, or a descendent of, type B in the type-system hierarchy. When the input capabilities of the selected component include multiple types, it is sometimes possible to satisfy these capabilities not only by the output capability of a single *AnnotationGroup* but also by a combination of output capability sets taken from several *AnnotationGroup* instances. The *ComparableAnalysisEngineAdapter* calculates all such possible combinations. It then creates a new CAS, and based on one calculated combination of *AnnotationGroup* instances, it populates the new CAS with a set of annotation instances from those *AnnotationGroup* instances in the original CAS. By creating a new CAS and copying only the required annotations, the amount of data stored in the CAS can be reduced, which increases performance when transmitting CAS to remote services.

The UIMA framework then calls the *newCasProduced* method of the flow controller to determine where to run the newly produced CAS. Our flow controller directs it to run the new CAS on the previously selected child component of which I/O capabilities had been passed to the *ComparableAnalysisEngineAdapter*.

The child component runs as if it were in a simple serial pipeline, using the newly created CAS and the copied annotations as its input. After running the child component, the flow controller specifies that the

ComparableAnalysisEngineAdapter should be run again. This time, the *ComparableAnalysisEngineAdapter* copies any outputs of the recently run child component satisfying its specified output capabilities back into the original CAS as part of a new *AnnotationGroup*. When copying these annotations, links to annotations that existed in the original CAS are relinked to the correct annotations in the original CAS. Once this is completed, the newly created CAS is discarded.

The *ComparableAnalysisEngineAdapter* then proceeds to another combination of *AnnotationGroup* instances satisfying the inputs of the selected child component, creates a new CAS, and repeats the aforementioned process. This is repeated with all combinations of *AnnotationGroup* calculated by the *ComparableAnalysisEngineAdapter* to meet the components’ input capabilities. The flow controller is then notified and selects another child component’s I/O capabilities to pass to the *ComparableAnalysisEngineAdapter*. This process also repeats until the flow controller has passed on the I/O capabilities of all of its child components.

Pluggable evaluation system

In addition to the virtual workflow architecture previously described, we need a mechanism to perform comparisons between the outputs of different virtual workflows. The problem is that there is a wide variety of comparison/evaluation metrics in NLP, even for a single type of annotation. Given a pair of annotation results to be compared, the comparison process can be generalized into the following steps.

1. Find a candidate list of annotations for each virtual workflow annotation result.
2. Compare candidate annotations one by one, deciding whether a pair of annotations is matched based on a specific metric.
3. Calculate comparison statistics.

These steps are usually manually implemented for each different comparison that is required. We propose a *pluggable* evaluation mechanism, where users only have to provide the aforementioned steps (or choose from our templates) and can directly plug in their own evaluation metrics as UIMA components.

By running a comparison workflow, we can obtain a set of *AnnotationGroup* instances that include the output types that we are interested in comparing. We require an *EvaluationIterator* component to be added to the end of the workflow. This is an aggregate component with an associate special flow controller that holds evaluation components as its children. These evaluation components each define an evaluation metric. The *EvaluationIterator* calculates which metrics can be applied for each

AnnotationGroup result, based on the input capabilities of its child evaluation components. Multiple evaluation metrics may be found to be applicable.

Then, *EvaluationIterator* stores pairs of these *AnnotationGroup* instances in a *ComparisonSet*-type annotation as *goldAnnotationGroup* and *testAnnotationGroup* fields. An evaluation component should calculate which pairs of annotations between *TestAnnotations* and *GoldAnnotations* are “matched” based on its own evaluation metric and store such *MatchedPairs* in the *ComparisonSet*. A visualization of this evaluation data is discussed later.

U-Compare platform

The simplest workflow creation system would only require users to specify which components to run and in which order. Our U-Compare platform aims to make this possible. We discuss features that are required to achieve this goal and then show our system, which allows users to create workflows by simply dragging and dropping components into place.

Interoperability for automation

First, in addition to the formal interoperability of UIMA, we need semantic interoperability of components, which corresponds to them having compatible type systems. If the components in a workflow are not type system compatible, they will not be able to build on each other’s results.

U-Compare defines its own type system [13] that satisfies the requirement for comparisons discussed earlier and contains most types commonly used in NLP.

Resource interoperability is also important. Resources not only should be compatible from the type-system point of view but also should be sufficiently reusable. If the I/O conditions of resources are complex, the probability of being able to combine arbitrary pairs of resources is low, making their reusability low. Thus, we claim that in order to improve reusability, resources should ideally have very simple I/O conditions—that is, simply a static list of I/O data types. During development, resources should be decomposed into as many different components as possible, with their I/O capabilities clearly specified. This would make it possible to calculate intercomponent dependences and improve reusability. U-Compare provides a component repository designed in such a manner, which is currently the world’s largest open collection of type-system-compatible UIMA components.

Resources are far less useful if they work only on specific platforms or in specific environments. Since both the Apache UIMA API and the U-Compare platform are implemented in pure Java, the entire platform is portable to any environment in which Java 6 is available. However, this is not sufficient for users because manual installation and configuration of resources is required, which may cause

problems due to platform misconfiguration and version mismatches. U-Compare provides a loader called UCLoader, which is a very small Java program that automatically downloads, installs, and updates the U-Compare platform and its resource files. UCLoader can be launched direct from a web browser via a link on the U-Compare website and saves users the potential problems caused by manual configurations. UCLoader caches resources on the local file system so that after it has been run once, an active Internet connection is no longer needed to run U-Compare.

The U-Compare platform provides easy access to components in the U-Compare component library. Components implemented in pure Java are locally deployed, and non-Java components are remotely deployed as web services. From a user’s perspective, remotely deployed components behave identically to locally deployed components. All components included with U-Compare can be used with other UIMA-based systems by adding our jar files to the classpath and making use of the UIMA descriptors provided with U-Compare.

U-Compare is fully UIMA compatible; the U-Compare type system and component repository are provided simply as a default toolkit.

Workflow creation GUI

In the main U-Compare view, the workflow creation panel, users can easily create UIMA CPE workflows by simply dragging and dropping desired components from the component library pane into the correct location in the workflow pane (see **Figure 4**). This view also allows users to edit and save component attributes including children of aggregates, to run workflows, and to export shareable components and workflow packages.

Third-party UIMA components can be easily added to the U-Compare component library. After installing a third-party component to the user’s machine, the user just has to add the required resource files to the U-Compare classpath via the “add to library” menu in the U-Compare workflow GUI. U-Compare automatically searches for UIMA component descriptors based on these classpath entries and gives users the choice to add any discovered components to the library. Once added, third-party components can be freely used in combination with native U-Compare components (type-system compatibility permitting).

The U-Compare GUI also provides support for the creation of the comparison workflows, as described earlier. When a parallel component is added to a workflow, the GUI internally switches to a comparison mode, automatically inserting the required annotation grouper and evaluation iterator components, and wrapping existing components into parallel components. Users only need to specify what types of annotations and which components they want to compare.

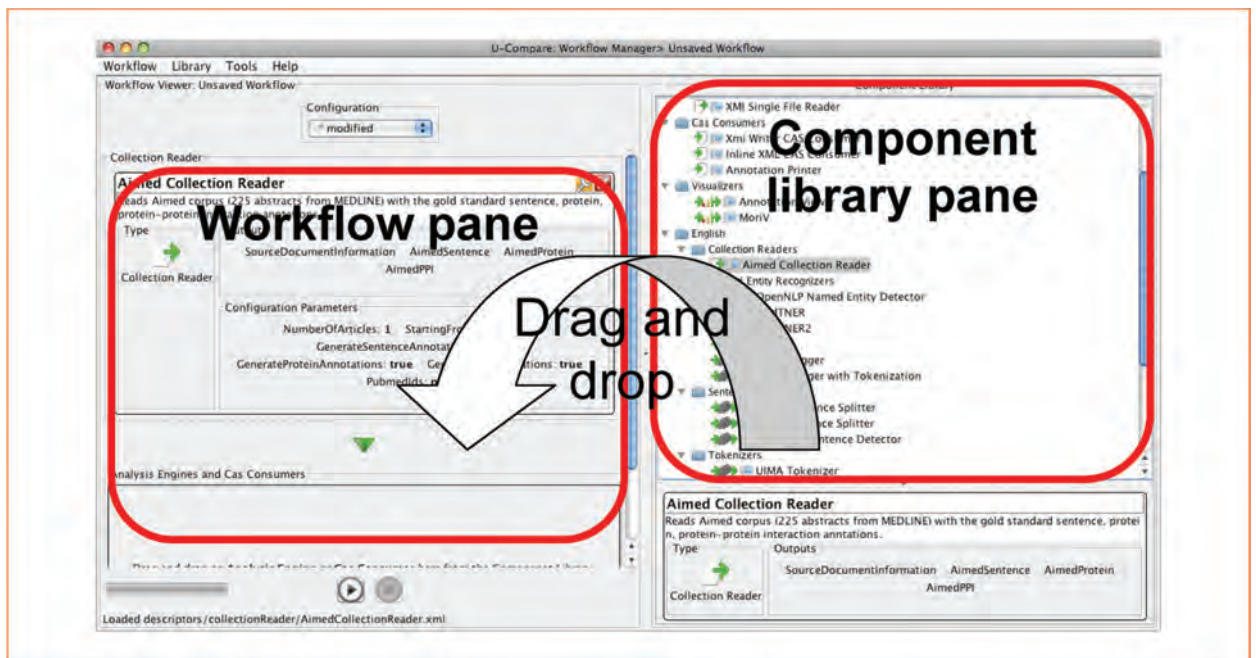


Figure 4
Screenshot of the workflow creation graphical user interface.

Statistics and visualizations

Since UIMA does not provide session handling features, sessions (results of workflow runs) must be handled outside of UIMA.

When a workflow is run from the workflow creation GUI, the statistics of the results of that session are automatically displayed. These statistics include runtime performance measures, individual and total annotations counts, and in the case of comparison workflows, precision, recall, and F scores. Workflows from saved sessions can also be loaded into the workflow configuration view.

In addition to numerical statistics, the U-Compare platform includes annotation visualization tools useful for NLP data (see **Figure 5**). These visualizations display annotations as underlines and arcs superimposed on the document text and easily highlight differences discovered during component comparison.

Other features and applications of U-Compare

UCLoader can run a specified workflow directly from the command line; this includes workflows created via the workflow creation GUI. Additionally, U-Compare provides UIMA components for standard I/O streams that communicate in a simple standoff annotation format. This allows easy embedding of workflows into other systems regardless of programming languages, as we did for Taverna [14].

We have built a distributed web service system that deploys any UIMA SOAP service to a cluster of machines parallelized, which behaves as a single web service component. U-Compare has been used in many tasks by both expert and nonexpert users, by individuals, and as part of worldwide challenges, such as protein-protein extraction from biological literature, BioNLP'09 shared task [15], Conference on Natural Language Learning (CoNLL) 2010 shared task [16], BioCreative II.5 [17], linkage with Taverna [14], and Chemistry Using Text Annotations (CheTA) [18].

Conclusion and future work

U-Compare is a modular NLP workflow creation system specifically designed to speed the process of creating, testing, and deploying NLP applications. It is built on a foundation of the UIMA framework and provides many of the tools necessary for NLP application development that UIMA currently lacks due to its more general nature. Those users not requiring this full set of capabilities will benefit from U-Compare's modular design, which allows many of its innovative features to be utilized independent of the platform as a whole. U-Compare has been already successfully used in a variety of practical applications where it has allowed users to achieve their goals with far less effort and with a reduced probability of user error.

As Apache UIMA now provides the UIMA Asynchronous Scaleout (UIMA-AS) API as its next-generation scalability

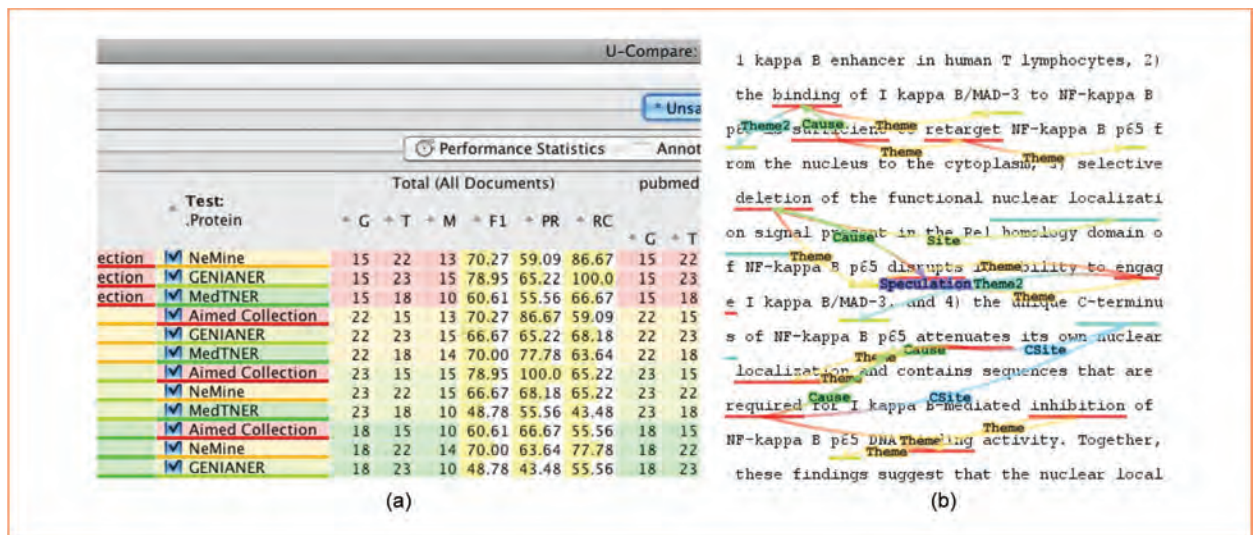


Figure 5

Screenshots of (a) comparison statistics between Almed corpus and three NERs. (b) U-Compare annotation viewer showing annotation spans as underlines and relations as arcs with relation labels, where different types are shown as different colors.

layer, migration from CPE to UIMA-AS and from SOAP to ActiveMQ services would be the next important step. UIMA-AS parallelization and our virtual workflow architecture do not conflict but are complementary. By integrating these two, it is possible to decrease the response time by processing parallel components physically in parallel.

Acknowledgments

The authors would like to thank Luke McCrohon of the University of Tokyo for the contributions to the graphical user interface designs and numerous valuable suggestions regarding this manuscript; Dr. Bill Black, Dr. Xinglong Wang, and Dr. BalaKrishna Kolluru of the National Centre for Text Mining (NaCTeM) for making their tools available in U-Compare; William Baumgartner and Chris Roeder of the University of Colorado for helping the type-system design and providing and maintaining their UIMA components. This work was supported in part by KAKENHI, Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan, under Grant 18002007 and Grant 21500130 and in part by Biotechnology and Biological Sciences Research Council (BBSRC) under Grant BB/F006012/1 and Grant BB/G013160/1. NaCTeM is supported by the Joint Information Systems Committee (JISC), U.K.

**Trademark, service mark, or registered trademark of Eclipse Foundation, Inc., or Sun Microsystems in the United States, other countries, or both.

References

1. D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn, "Taverna: A tool for building and running workflows of services," *Nucleic Acids Res.*, vol. 34, no. Suppl 2, pp. W729-W732, Jul. 1, 2006.
2. D. Blankenberg, G. Von Kuster, N. Coraor, G. Ananda, R. Lazarus, M. Mangan, A. Nekrutenko, and J. Taylor, "Galaxy: A web-based genome analysis tool for experimentalists," *Curr. Protocols Mol. Biol.*, vol. Chapter 19, pp. 19.10-19.10-21, Jan. 2010.
3. I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock, "Kepler: An extensible system for design and execution of scientific workflows," in *Proc. 16th Int. Conf. Sci. Stat. Database Manage.*, 2004, pp. 423-424.
4. H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, "GATE: A framework and graphical development environment for robust NLP tools and applications," in *Proc. 40th Anniv. Meeting Assoc. Comput. Linguistics*, 2002, pp. 168-175.
5. D. Ferrucci, A. Lally, D. Gruhl, E. Epstein, M. Schor, J. W. Murdock, A. Frenkiel, E. W. Brown, T. Hampp, Y. Doganata, C. Welty, L. Amini, G. Kofman, L. Kozakov, and Y. Mass, "Towards an interoperability standard for text and multi-modal analytics," IBM, Yorktown Heights, NY, Res. Rep. RC24122, 2006.
6. Apache UIMA. [Online]. Available: <http://uima.apache.org/>
7. OASIS UIMA TC. [Online]. Available: <http://www.oasis-open.org/committees/uima/>
8. W. A. Baumgartner, Jr., K. B. Cohen, and L. Hunter, "An open-source framework for large-scale, flexible evaluation of biomedical text mining systems," *J. Biomed. Discov. Collab.*, vol. 3, no. 1, p. 1, Jan. 2008, DOI: 10.1186/1747-5333-3-1.
9. U. Hahn, E. Buyko, R. Landefeld, M. Mühlhausen, M. Poprat, K. Tomanek, and J. Wermter, "An overview of JCoRe, the JULIE Lab UIMA component repository," in *Proc. LREC Workshop, Towards Enhanc. Interoperability Large HLT Syst.: UIMA NLP*, Marrakech, Morocco, 2008, pp. 1-8.
10. Open Health Natural Language Processing (OHNLP) Consortium. [Online]. Available: <http://cabig-kc.nci.nih.gov/Vocab/KC/index.php/OHNLP>

11. N. Hernandez, F. Poulard, M. Vernier, and J. Rocheteau, "Building a French-speaking community around UIMA, gathering research, education and industrial partners, mainly in Natural Language Processing and Speech Recognizing domains," in *Proc. LREC Workshop New Challenges NLP Frameworks*, Valletta, Malta, 2010. [Online]. Available: <http://hal.archives-ouvertes.fr/hal-00481459/en/>
12. *U-Compare: Share and Compare Tools With UIMA*. [Online]. Available: <http://u-compare.org/>
13. Y. Kano, L. McCrohon, S. Ananiadou, and J. Tsujii, "Integrated NLP evaluation system for pluggable evaluation metrics with extensive interoperable toolkit," in *Proc. SETQA-NLP Workshop, NAACL-HLT*, Boulder, CO, 2009, pp. 22–30.
14. Y. Kano, P. Dobson, M. Nakanishi, J. Tsujii, and S. Ananiadou, "Text mining meets workflow: Linking U-Compare with Taverna," *Bioinformatics*, vol. 26, no. 19, pp. 2486–2487, Oct. 2010.
15. J. D. Kim, T. Ohta, S. Pyysalo, Y. Kano, and J. Tsujii, "Overview of BioNLP'09 shared task on event extraction," in *Proc. BioNLP Workshop Companion Volume Shared Task*, Boulder, CO, 2009, pp. 1–9.
16. R. Farkas, V. Vincze, G. Móra, J. Csirik, and G. Szarvas, "The CoNLL 2010 shared task: Learning to detect hedges and their scope in natural language text," in *Proc. 14th Conf. CoNLL: Shared Task*, Uppsala, Sweden, 2010, pp. 1–12.
17. R. Sætre, K. Yoshida, M. Miwa, T. Matsuzaki, Y. Kano, and J. Tsujii, "AkaneRE relation extraction: Protein interaction and normalization in the BioCreative II.5 challenge," in *Proc. BioCreative II.5 Workshop Special Session| Digit. Annotations*, Madrid, Spain, 2009, p. 33.
18. *CheTA: Chemistry Using Text Annotations*. [Online]. Available: <http://nactem.ac.uk/cheta/>

Received July 31, 2010; accepted for publication
December 15, 2010

Yoshinobu Kano *Department of Computer Science, University of Tokyo, Bunkyo-ku, Tokyo 113-0033, Japan (kano@is.s.u-tokyo.ac.jp)*. Mr. Kano received a B.S. degree in physics in 2001 and an M.S. degree in information science and technology in 2003, both from the University of Tokyo. He is a Research Associate in the Research Group led by Professor J. Tsujii at the University of Tokyo. He has been leading the U-Compare initiative designing and implementing the U-Compare system as the technical lead. He received an IBM Innovation Award in 2008.

Makoto Miwa *Department of Computer Science, University of Tokyo, Bunkyo-ku, Tokyo 113-0033 Japan (mmiwa@is.s.u-tokyo.ac.jp)*. Dr. Miwa received the M.Sc. and Ph.D. degrees in science from the University of Tokyo, Japan, in 2005 and 2008, respectively. He has since been working as a researcher at the University of Tokyo, under the supervision of Professor J. Tsujii. His research interests include machine learning in natural language processing for biomedical texts (BioNLP) and computer games.

Kevin Bretonnel Cohen *Computational Bioscience Program, University of Colorado School of Medicine, Aurora, CO 80045 USA (Kevin.Cohen@gmail.com)*. Mr. Cohen is the Biomedical Text Mining Group Lead of the Center for Computational Pharmacology at the University of Colorado School of Medicine. He is also the Lead Artificial Intelligence Engineer at the MITRE Corporation, Human Language Technology Division. He is a popular speaker at biomedical text mining and computational biology events and is the chairman of the Association for Computational Linguistics' SIGBIOMED special interest group on biomedical natural language processing.

Lawrence E. Hunter *Computational Bioscience Program, University of Colorado School of Medicine, Aurora, CO 80045 USA (Larry.Hunter@ucdenver.edu)*. Dr. Hunter received a B.A. degree in psychology from Yale University in 1982, and M.S. and Ph.D. degrees

in computer science from Yale University in 1987 and 1989, respectively. He is Professor and Director of the Computational Bioscience Program at the University of Colorado School of Medicine. He subsequently joined the scientific staff at the U.S. National Library of Medicine (National Institutes of Health [NIH]) where in 1994 he received the Regents Award for Scholarship and Technical Achievement. In 1999, he joined the National Cancer Institute (NIH) as chief of the Section on Biostatistics and Informatics and in 2000 joined the faculty of the University of Colorado School of Medicine, where he has been working on computational methods for analysis of genome-scale molecular biology data. In 2003, the American Association for Artificial Intelligence presented him with the Englemore Prize for innovative applications of artificial intelligence. He is the author of two patents and more than 100 peer-reviewed scientific papers. He is a fellow of the American College of Medical Informatics and a fellow of the International Society for Computational Biology.

Sophia Ananiadou *School of Computer Science, University of Manchester and National Centre for Text Mining, M1 7DN, UK (sophia.ananiadou@manchester.ac.uk)*. Professor Ananiadou is Director of the National Centre for Text Mining (NaCTeM), and Professor in Text Mining in the School of Computer Science, University of Manchester. Her current research includes terminology management, text mining-based visualization of biochemical networks, data integration using text mining, text mining interoperability, and building biomedical terminologies and automatic event extraction of bioprocesses. She has been awarded the 2004 Daiwa Adrian prize and the IBM UIMA innovation award (2006, 2007, and 2008) for her leading work on text mining interoperability in biomedicine. She has more than 160 publications in journals, conferences, and books. Her funding comes from EU, JISC, BBSRC, National Institutes of Health, and industry.

Jun'ichi Tsujii *Department of Computer Science, University of Tokyo, Bunkyo-ku, Tokyo 113-0033 Japan (tsujii@is.s.u-tokyo.ac.jp)*. Dr. Tsujii received the B.Eng., M.Eng., and Ph.D. degrees in electrical engineering from Kyoto University, Japan, in 1971, 1973, and 1978, respectively. He was an assistant professor and associate professor at Kyoto University, before taking up the position of professor of computational linguistics at the University of Manchester Institute for Science and Technology (UMIST) in 1988. Since 1995, he has been a professor in the Department of Computer Science at the University of Tokyo. He is also a professor of text mining at the University of Manchester (half-time) and the research director of the U.K. National Centre for Text mining (NaCTeM) since 2004. He was the president of the Association for Computational Linguistics (ACL) in 2006 and is a permanent member of the International Committee on Computational Linguistics (ICCL) since 1992. He received an IBM Science Award in 1988, an IBM Faculty Award in 2005, and the Medal of Honor with Purple Ribbon by the Government of Japan in 2010.