POWER5 system microarchitecture

B. Sinharoy
R. N. Kalla
J. M. Tendler
R. J. Eickemeyer
J. B. Joyner

This paper describes the implementation of the IBM POWER5™ chip, a two-way simultaneous multithreaded dual-core chip, and systems based on it. With a key goal of maintaining both binary and structural compatibility with POWER4™ systems, the POWER5 microprocessor allows system scalability to 64 physical processors. A POWER5 system allows both single-threaded and multithreaded execution modes. In single-threaded execution mode, a POWER5 system allows for higher performance than its predecessor POWER4 system at equivalent frequencies. In multithreaded execution mode, the POWER5 microprocessor implements dynamic resource balancing to ensure that each thread receives its fair share of system resources. Additionally, software-settable thread priority is enforced by the POWER5 hardware. To conserve power, the POWER5 chip implements dynamic power management that allows reduced power consumption without affecting performance.

Introduction

IBM introduced the POWER4* systems in 2001 [1]. Incorporating many features previously included only in mainframe systems, POWER4 systems also included unique packaging and system architecture. A POWER4 chip integrates onto a single die two processor cores, a shared second-level cache, a directory for an off-chip third-level cache, and the circuitry necessary to interconnect it with other POWER4 chips to form a system. In addition to the thread-level parallelism inherent in the dual-processor chip, high-instruction-level parallelism was achieved through an out-of-order execution design.

The POWER5* chip is the next-generation chip. In designing the POWER5 system, a key goal was to maintain both binary and structural compatibility with existing POWER4 systems to ensure not only that binaries would continue to execute properly, but that all application optimizations would carry forward to newer systems. With that as a base requirement, we specified increased performance and other server functional enhancements in the areas of server virtualization, reliability, availability, and serviceability at both the chip and system levels.

In this paper, we describe the approach used in improving performance. We enhanced thread-level parallelism by allowing two threads, or instruction streams, to execute on each of the two processor cores. We first present background information on different multithreaded hardware implementations. We then provide a brief overview of the POWER5 structure and contrast it to a POWER4 system. With that as background, we describe the POWER5 simultaneous multithreading (SMT) implementation in detail. We then describe the memory subsystem and how systems using more than a single POWER5 chip are formed. As SMT makes greater use of chip resources, it consumes greater power. To counteract this, the POWER5 chip incorporates dynamic power management into its design to allow the processor to save switching power when it is not needed without affecting performance. This capability is also described. Finally, the reliability, availability, and serviceability (RAS) enhancements in POWER5 systems are briefly described.

Multithreading background

Typical advanced microprocessors have executed instructions from a single instruction stream. Performance has improved over the years through many architectural techniques, such as caches, branch prediction, and out-of-order execution. These lead to improved performance at a given processor frequency by increasing instruction-level parallelism. At the same time, through the use of longer pipelines and fewer logic levels per stage, processor frequencies have been increasing more rapidly than the technology. Despite the

0018-8646/05/$5.00 © 2005 IBM

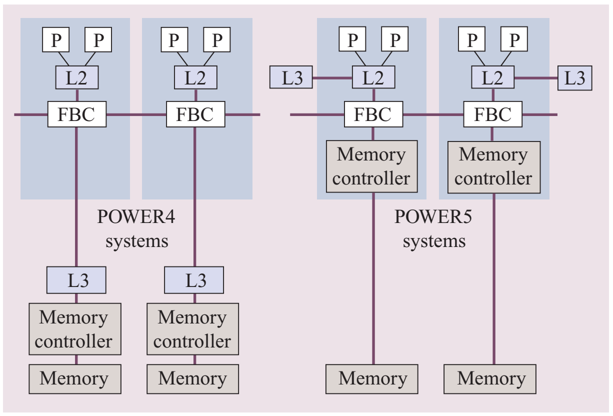

Figure 1

Modifications to POWER4 system structure. (P: processor; FBC: fabric bus controller; L2, L3: Level 2, Level 3 caches.) Reprinted with permission from [2].

architectural advances, the frequency improvements lead to lower execution unit utilizations. This is due to an increase in the number of cycles for instruction execution, cache misses, branch mispredicts, and memory access. It is common to see average execution unit utilizations of 25% across a broad range of workloads. To increase execution unit utilization, multithreading has been introduced. This creates thread-level parallelism that increases processor throughput. To the operating system, multithreading looks almost the same as symmetric multiprocessing. There are at least three different methods for handling multiple threads: coarse-grain multithreading, fine-grain multithreading, and simultaneous multithreading.

In coarse-grain multithreading, only one thread executes at any given instant in time. When a thread encounters a long-latency event, such as a cache miss, the hardware swaps in a second thread to use the machine resources rather than letting it idle. By allowing other work to use what otherwise would have been idle cycles, overall system throughput is increased. To conserve chip area, both threads share many of the system resources, such as architected registers. Hence, to swap program control from one thread to another requires several cycles. IBM introduced coarse-grain threading on the IBM pSeries* S85 [3].

Fine-grain multithreading switches between threads each cycle. In this class of machines [4], a different thread is executed in a round-robin fashion. As in coarse-grain multithreading, the architected states of multiple threads are all maintained in the processor. Fine-grain multithreading allows overlap of short pipeline latencies by letting another thread fill in execution gaps that would otherwise exist. With a larger number of threads, longer latencies can be successfully overlapped. For long-latency events in a single thread, if the number of threads is less than the number of latency cycles, there will be empty execution cycles for that thread. To accommodate this design, hardware facilities are duplicated. When a thread encounters a long-latency event, its cycles remain unused.

Simultaneous multithreading maintains the architected states of multiple threads. This type of multithreading is distinguished by having the ability to schedule instructions from all threads concurrently [5]. On any given cycle, instructions from one or more threads may be executing on different execution units. With SMT, the system adjusts dynamically to the environment, allowing instructions to execute from each thread if possible while allowing instructions from one thread to utilize all of the execution units if the other thread(s) cannot make use of them. This allows the system to dynamically adjust to the environment. The POWER5 system implements two threads per processor core. Both threads share execution units if both have work to do. If one thread is waiting for a long-latency event, the other thread can achieve a greater share of execution unit time.

POWER5 system structure

Figure 1 shows a high-level system structure for POWER4 and POWER5 systems. POWER4 systems were designed to handle up to 32 physical processors on 16 chips. Going beyond 32 processors can increase interprocessor communication, resulting in higher traffic on the interconnection fabric; this can cause greater contention and can affect system scalability negatively. In POWER5 systems, by moving the L3 cache from the memory side of the fabric to the processor side of the fabric, we are able to more frequently satisfy L2 misses with hits in the 36-MB off-chip L3, thus eliminating L3 hit traffic from the interchip buses. The L3 operates as a victim cache[1] for the L2, with separate 16-byte-wide buses for reads and writes that operate at half processor speed. Data is staged to the L3 only when it is replaced from the L2. Similarly, references to data in the L3 cause that cache line to be reloaded into the L2. Only modified data that is replaced from the L3 is written back, or cast-out to memory. Unmodified data that is replaced in the L3 is discarded.

In the POWER4 system, the L3 cache is in the path between the processor chip and the memory controller. In the POWER5 system, with the L3 removed from this path, we were also able to move the memory controller on-chip. These two changes also had significant additional benefits: reduced latency to the L3 and to memory, increased bandwidth to the L3 cache and the memory, and improvement in intrinsic reliability resulting from the reduction in the number of chips necessary to build a

[1]When a new cache line is written in the L2 cache, it often replaces (or victimizes) a valid cache line. The line being replaced in the L2 cache is written in the L3 cache.

system. The 32-MB L3 in a POWER4 system comprises two chips; a POWER5 system uses one chip for the 36-MB L3.

POWER5 chip overview

The POWER5 processor implements the 64-bit PowerPC* architecture. Inside the chip, shown in **Figure 2**, two identical processor cores are contained on a single die. The processor cores each support two logical threads. To the operating system, the chip appears as a four-way symmetric multiprocessor. A 1.875-MB L2 cache is shared by the two processor cores. There are three partitions, or slices, of the L2, each of which is ten-way set-associative, with 512 congruence classes of 128-byte lines. The slices are identical, with separate controllers for each. The real address determines the slice of the L2 in which the cache line can be found. This is done with modulo 3 arithmetic on a large number of the real address bits such that each real address selects a unique slice, spreading L2 accesses more uniformly across the slices. The L2 controller can be independently accessed by either processor core. The L3 directory for the off-chip 36-MB L3 is also integrated onto the POWER5 chip. Having the L3 directory on-chip allows the directory to be checked after an L2 miss without experiencing off-chip delays. The L3 is also implemented as three slices, with each slice acting as a victim cache for one of the L2 slices. Each slice is 12-way set-associative, with 4,096 congruence classes of 256-byte lines managed as two 128-byte sectors to match the L2 line size. To reduce memory latencies, the memory controller is integrated onto the POWER5 chip, eliminating driver and receiver delays to an external controller.

For each read and write L2 cache operation, a separate cache coherency engine is used to control the state of the line until the operation is complete. The POWER5 design increases the number of coherency engines and the number of store queue entries in the L2 cache controller to double those in a POWER4 system. This improves SMT performance and supports 64-way symmetric multiprocessing.

Accesses between the POWER5 chip and the L3 are across two unidirectional 16-byte-wide buses operating at half processor frequency. Access between memory and the on-chip memory controllers is via two unidirectional buses operating at twice the dual in-line memory module (DIMM) frequency.[2] The data memory read bus is 16 bytes wide, while the write memory bus is 8 bytes wide.

The POWER5 design includes dynamic power management and enhancements to the reliability, availability, and serviceability (RAS) attributes of the system. These areas are discussed later.

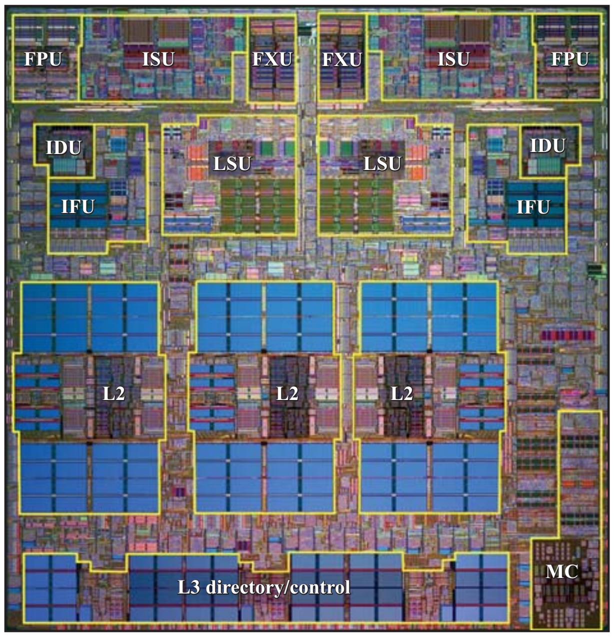

Figure 2

POWER5 die photo showing major chip components. (FXU: fixed-point execution unit; ISU: instruction sequencing unit; IDU: instruction decode unit; LSU: load/store unit; IFU: instruction fetch unit; FPU: floating-point unit; MC: memory controller.) Reprinted with permission from [2].

The POWER5 chip uses silicon-on-insulator (SOI) devices and copper interconnects. SOI technology reduces device capacitance to increase transistor performance. Copper interconnects decrease wire resistance and reduce delays in wire-dominated chip timing paths. The chip has eight levels of metal and measures 389 mm^2.

Both the POWER4+* chip[3] and the initial POWER5 chip are manufactured using a 130-nm process. Chip layouts are very similar. Chip size has increased from 267 mm^2 for the POWER4+; chip to 389 mm^2 in the POWER5 chip, corresponding to an increase from 184 million transistors to 276 million transistors. The additional growth in chip area is summarized in **Table 1**. The POWER5 chip contains 5,370 I/O pins comprising 2,313 signal I/Os and 3,057 power I/Os. The distribution of POWER5 signal I/Os by major function is shown in **Table 2**.

POWER5 processor core

The POWER5 processor core is designed to support both enhanced simultaneous multithreaded and single-

[2]Currently 266-MHz DDR1 or 533-MHz DDR2 DIMMs are available. Depending on the POWER5 system model, one or both of these memory options are supported.

[3]The initial POWER4 chip was fabricated using a 180-nm process. The POWER4+ chip, first introduced in late 2002, is architecturally similar to the POWER4 chip but is fabricated using a 130-nm process. Comments with respect to the POWER4 chip are also completely applicable to the POWER4+ chip.

Table 1 Increase in POWER5 chip area over that of the POWER4+ chip.

Chip area increase from POWER4+		*Primary reason*
Component	*Percentage*	
Core	10	Addition of SMT
L2 cache	11	Increase in size from 1.5 MB to 1.875 MB and increase in coherency engines
L3 directory and control	7	Increase in directory size to accommodate larger capacity and smaller L3 line size
Memory controller	3	New to POWER5 chip
Fabric controller	5	Enhancements to distributed switch
I/Os	5	Increase in bus speeds
Other	5	
Total	46	

Table 2 Distribution of 2,313 POWER5 signal I/Os by major function.

Function	*Signal I/Os* (%)
SMP fabric	60
L3 and memory buses	32
I/O bus	4
Clocks	4

threaded (ST) modes of operation. The POWER5 instruction pipeline shown in **Figure 3(a)** is identical to the POWER4 instruction pipeline. All pipeline latencies, including the branch misprediction penalty and load-to-use latency with an L1 data cache hit in POWER5, are the same as in POWER4. The POWER5 instruction data flow is shown in **Figure 3(b)**.

Instruction pipeline

After loading the program counter with the address for the next instruction [instruction fetch (IF) pipeline stage in Figure 3(a)], up to eight instructions can be fetched from the instruction cache (IC pipeline stage) every cycle. The instruction cache and the instruction translation facility are shared between the two threads. In a given cycle, instructions are fetched from the same thread. Instructions are then scanned for branches and if a branch is found, the direction of that branch is predicted (BP pipeline stage) using three branch history tables (BHTs) that are shared by the two threads. Two of the BHTs are used for predicting branch directions on the basis of bimodal and path-correlated branch-prediction mechanisms. The third BHT is used to predict which of these prediction mechanisms is more likely to predict the correct direction. If the instructions fetched are all branches, all can be predicted at the same time. In addition to the direction prediction, the POWER5 microprocessor also predicts the target of a taken branch in this group of eight instructions. Branch target addresses for the PowerPC branch to link register (*bclr*) and branch to count register (*bcctr*) instructions can be predicted using a hardware-implemented return address stack and a count cache mechanism, respectively. Target addresses for absolute and relative branches are computed directly as part of the branch scan function. If there is a taken branch, the program counter is loaded with the target address of the branch. Otherwise, the program counter is loaded with the address of the next sequential instruction from which fetching is being done. Each branch is entered in the branch information queue (BIQ) at instruction fetch time. The queue saves the necessary information to recover from a mispredicted branch. Entries are deallocated in program order when branches are executed.

In SMT mode, two separate program counters are used, one for each thread. Instruction fetches alternate between the two threads. Similarly, branch prediction alternates between threads. In ST mode, only one program counter is used, and instructions can be fetched for that thread every cycle.

After fetching (before pipeline stage D1), instructions are placed in separate instruction buffers for the two threads. These buffers contain 24 instructions each, slightly smaller than the single queue in a POWER4 microprocessor. Though smaller, the queue management logic was changed to make it more efficient. On the basis of thread priority (described below), up to five instructions are fetched from one of the instruction buffers (D0 pipeline stage), and a group is formed (pipeline stages D1 through D3). Instructions in a group are all from the same thread. All instructions in the group are decoded in parallel.

When all of the resources necessary for dispatch are available for the group, the group is dispatched (GD pipeline stage). Instructions flow between group formation and dispatch in program order (D0 through GD pipeline stages). After dispatch, each instruction flows through the register-renaming facilities, where the logical register numbers in the instruction are mapped to physical registers (MP pipeline stage). The register files are dynamically shared by the two threads. In ST mode, all physical registers are available to the single

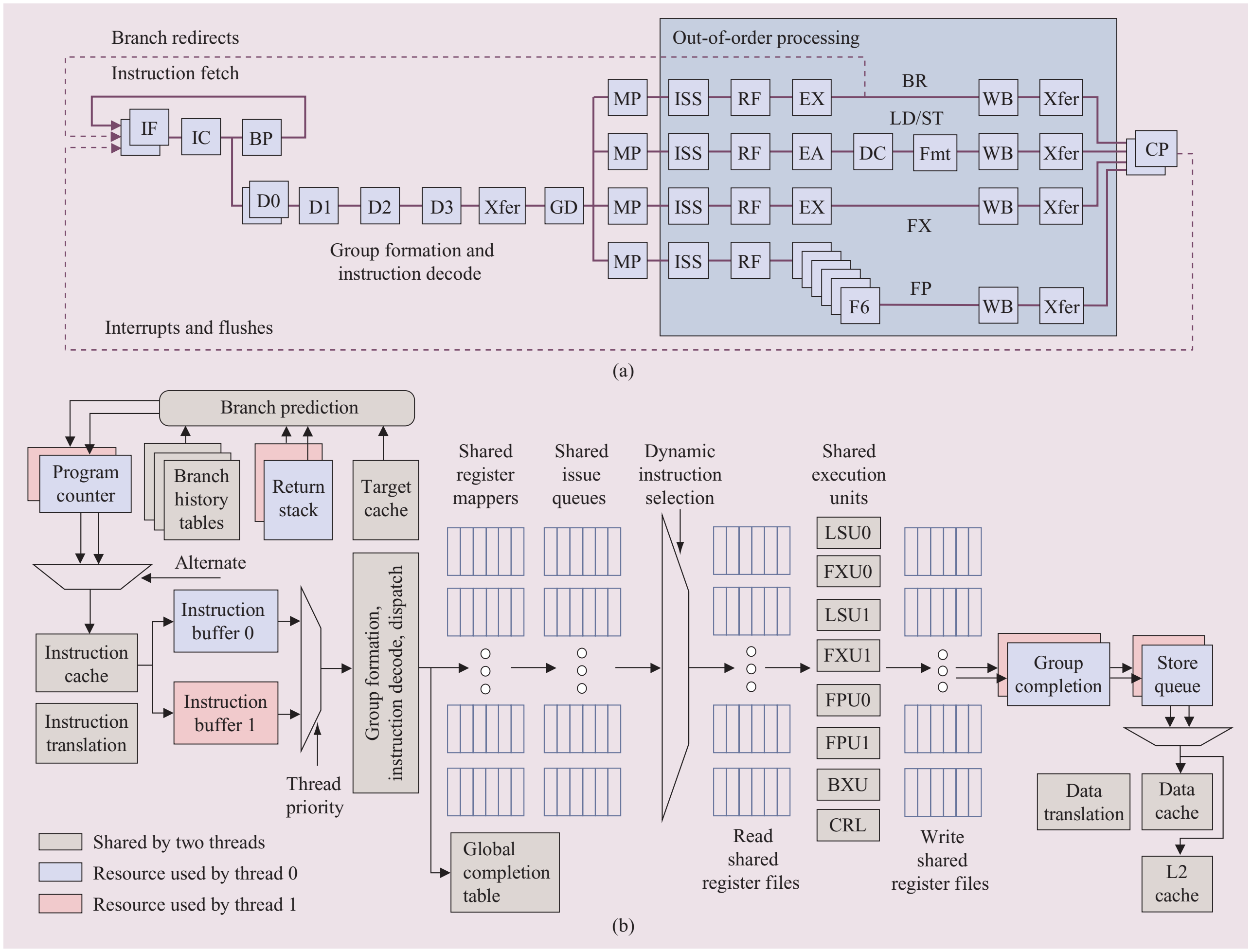

Figure 3

POWER5 (a) instruction pipeline; (b) instruction data flow. (IF: instruction fetch; IC: instruction cache; BP: branch predict; D0: decode stage 0; Xfer: transfer; GD: group dispatch; MP: mapping; ISS: instruction issue; RF: register file read; EX: execute; EA: compute address; DC: data caches; F6: six-cycle floating-point execution pipe; Fmt: data format; WB: write back; CP: group commit; BXU: branch execution unit; CRL: condition register logical execution unit.) Reprinted with permission from [2].

thread, allowing higher instruction-level parallelism. After register renaming, instructions are placed in shared issue queues.

To simplify the logic for tracking instructions through the pipeline, instructions are tracked as a group. Control information for each group of dispatched instructions is placed in a global completion table (GCT) at the time of dispatch [1]. The GCT entry contains the address of the first instruction in the group. Logically, the entries in the GCT are allocated in program order for each thread. As instructions finish execution, that information is registered in the GCT entry for the group. An entry is de-allocated from the GCT when the group is committed. While the entries in the GCT are allocated and deallocated in program order for a given thread, the entries can be intermingled between the two threads in any arbitrary order.

In addition to allocating GCT and register renaming, other necessary conditions for dispatch are to allocate load reorder queue (LRQ) and store reorder queue (SRQ) entries for the load and store instructions in the group. These two queues maintain the program order of loads and stores within a thread and allow for checking of address conflicts between loads and stores.

When all input operands for an instruction are available, it becomes eligible for issue. Among the eligible instructions in the issue queue, one of the oldest is selected and issued for execution (ISS pipeline stage). For instruction issue, no distinction is made between instructions from the two threads. There is no priority difference between the threads, and instruction issue is independent of the GCT group to which the instruction belongs; hence, instructions can issue concurrently from multiple groups. Up to eight instructions, one to each execution unit, can issue in a cycle. When issued, the instruction reads its input physical registers (RF pipeline stage), executes on the proper execution unit (EX pipeline stage for the branch execution unit, the fixed-point execution units, and the logical condition register unit; EA, DC, and Fmt pipeline stages for the load/store units; and F1 through F6 pipeline stages for the floating-point execution units); and writes the result back to the output physical register (WB pipeline stage).

When all of the instructions in a group have executed (without generating any exception) and the group is the oldest group of a given thread, the group is committed (CP pipeline stage). One group can commit per cycle from each thread.

Enhancements to make best use of simultaneous multithreading

The POWER5 SMT implementation required changing certain resources to accommodate the second thread of instructions; in order to enhance performance, additional capabilities were added. This section discusses the changes to the POWER4 design introduced with the POWER5 microprocessor specifically due to the addition of SMT.

Resource sizes

To efficiently support SMT, all resources have been tuned for improved performance within an area and power budget constraint. A detailed performance model was used throughout the design to study and tune resources for these design tradeoffs. The first-level instruction and data caches are the same size (64 KB and 32 KB, respectively) as in POWER4 systems, but their associativity has been doubled to two-way and four-way, respectively. Instruction and data cache entries can be fully shared between threads.

In the PowerPC architecture, address translation is performed in two steps. First, the effective address is translated to the virtual address. In the POWER5 processor, the segment table is cached in a fully associative 64-entry segment lookaside buffer (SLB), one per thread. Next, the virtual address is translated to the real address using a hashed page table that is also maintained in memory. In the POWER5 processor, the page table is cached in a 1,024-entry, four-way set-associative translation lookaside buffer (TLB). To facilitate fast translation, two first-level translation tables are used, one for instructions and one for data, to provide a fast, effective address to a real address translation. For most accesses, address translation is satisfied with a translation hit in the first-level translation tables. The SLB and TLB are looked up only if the translation cannot be accomplished using the first-level translation tables. The first-level data translation table is a fully associative 128-entry array. The first-level instruction translation table is a two-way set-associative 128-entry array. Entries in both first-level translation tables are tagged with the thread number and are not shared between threads. The TLB organization was not changed in POWER5 systems. Entries can be shared between threads. The BHT and count cache are also unchanged to support SMT. The return address stack was duplicated, since it contains ordering that must be maintained within a thread. The four instruction prefetch buffers are split between the two threads; each thread can independently process instruction cache misses and instruction prefetches.

The size and structure of some of the queues have been changed for POWER5 systems to enhance SMT operation balancing performance, power, area, and complexity. The GCT is the main control point of the out-of-order processor. The GCT design was changed to support SMT by implementing it as a linked list so that each thread can be independently allocated and de-allocated. A fully shared GCT allows the number of entries to remain the same as in POWER4 systems. Register renaming is changed only slightly from the POWER4 implementation. Each logical register number has a thread bit appended, and these are then mapped as usual. Because the second thread comes with its set of architected registers, the number of register renames of each type was increased, with only the FPSCR and the link/count registers excepted. Unlike the other registers, an FPSCR is associated with each GCT entry. Because the number of GCT entries was not increased, there was no need to increase the FPSCR rename registers. Extensive modeling and experience gained with POWER4 systems indicated that there was no need to increase the size of the link/count rename registers. In POWER5 systems, the size of the issue queues remains the same as in POWER4 systems, with the exception of the floating-point issue queues, which were increased from a total of 20 entries to 24. **Table 3** summarizes the rename registers and issue queue sizes for both POWER4 and POWER5 processors.[4]

In the POWER4 design, both the LRQ and the SRQ contain 32 entries. Entries are allocated and de-allocated in program order. They are allocated at dispatch. The LRQ is de-allocated at completion, and the SRQ is de-allocated after completion once the store has been sent to the L2. For SMT, program order must be maintained

[4]The table shows four more GPRs and one more 4-bit field for the CR than architected in the PowerPC architecture. These additional registers are used by the system to handle cracked and microcoded instructions, instructions that are expanded by the hardware for ease in implementation.

Table 3 Rename registers and issue queue sizes.

Resource type	*Logical size (per thread)*	*Physical size*	
		POWER4	*POWER5*
GPRs	32 (+4)	80	120
FPRs	32	72	120
CRs[†]	8 (+1) 4-bit fields	32	40
Link/count registers	2	16	16
FPSCR[†]	1	20	20
XER[†]	Four fields	24	32
Fixed-point and load/store issue queue	Shared by both threads	36	36
Floating-point issue queue	Shared by both threads	20	24
Branch execution issue queue	Shared by both threads	12	12
CR logical issue queue	Shared by both threads	10	10

[†]CR is an acronym for *condition register*. Architecturally, the CR consists of eight 4-bit fields that indicate how to resolve the direction of a branch instruction. In POWER4 and POWER5 systems, each field of the CR is treated as a separate register. FPSCR is an acronym for *floating-point status and control register*. XER is an acronym for *fixed-point exception register*. Four of the XER fields are renamed in POWER4 and POWER5 systems. See [1, 6] for additional details.

within a thread, but the threads must be able to independently allocate and de-allocate entries. Because the address checking to ensure memory consistency occurs on a thread basis, it was simpler to split the LRQ and SRQ into two halves, one per thread; this resulted in cases in which one thread could run out of LRQ or SRQ entries. Rather than increase the physical size of these queues, each was extended by providing 32 additional virtual queue entries, 16 per thread. A virtual queue entry contains sufficient information to identify the instruction but not the address specified for the load or store instruction or the data to be stored for a store instruction. This mechanism provides a low-cost method of extending the LRQ and SRQ sizes and not stalling instruction dispatch. At instruction dispatch, if a real LRQ or SRQ entry is not available and a virtual entry is available, the instruction can dispatch with the virtual entry. As real entries become available, virtual entries are converted to real entries, with the virtual queue entry returned to the pool for possible use by younger instructions. Before a load or a store can issue, its LRQ or SRQ entry, respectively, must be associated with a real entry.

The size of the BIQ remains at 16 entries as in POWER4 systems. In SMT mode it is split in two, with eight entries per thread. At the front end of the pipeline, instruction groups typically alternate between threads. Performance modeling showed that a split BIQ would be sufficient for SMT performance. Other queues were similarly studied to determine any changes needed to support SMT. The POWER4 eight-entry load miss queue (LMQ) was changed in the POWER5 design by adding a thread bit that allows the LMQ to be dynamically shared between the two threads.

Dynamic resource balancing

In a multithreaded system, one thread may use a significant amount of system resources, potentially blocking other threads. To prevent this, resource-balancing logic has been included in the POWER5 design to keep two threads with different processing requirements flowing well through the system. Dynamic resource-balancing logic monitors resources, such as the GCT and the LMQ, to determine whether one thread exceeds a threshold. If that occurs, the progress of the offending thread is throttled back, allowing the sibling thread to flow through the machine without encountering congestion. As an example, if one thread encounters multiple L2 cache misses, dependent instructions can back up in the issue queues. This can inhibit the dispatching of additional groups, causing the second thread to slow down. Resource-balancing logic detects the point at which a thread reaches a threshold of load misses in the L2 cache and translation misses in the TLB. When this happens, thread performance is throttled. Similarly, resource-balancing logic monitors the GCT to determine the number of entries each thread is using. If the balancing logic detects that one thread is beginning to use too many GCT entries, potentially blocking the other thread, it throttles back the thread that is using excessive GCT entries.

Depending on the situation, the POWER5 microprocessor employs one of three mechanisms to throttle threads:

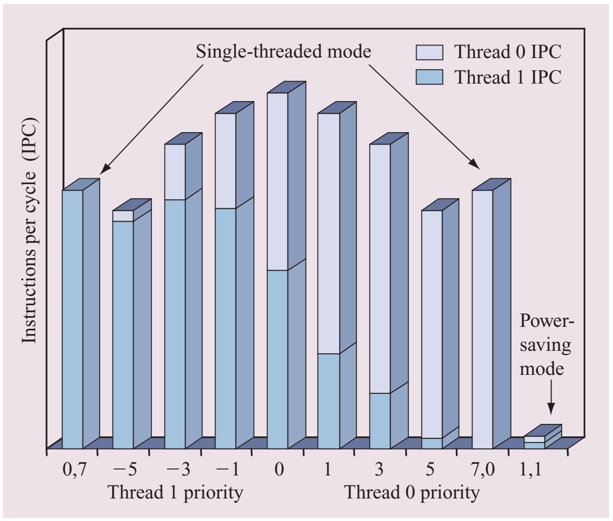

Figure 4

Conceptual view of the effects of priority on performance. Reprinted with permission from [2].

- *Reducing the priority of the thread* is the primary mechanism for situations in which a thread uses more than a predetermined number of GCT entries. (Each POWER5 core permits eight priority levels for each thread. Priority levels are set by software and enforced by the hardware. The hardware may temporarily adjust the priority of a thread to throttle its execution. Thread priority is discussed more fully in the next section.)
- *Inhibiting the instruction decoding of the thread until the congestion clears* is the primary mechanism for throttling a thread that incurs a specified number of L2 cache misses.
- *Flushing all of the thread instructions that are waiting for dispatch and stopping the thread from decoding additional instructions until the congestion clears* is the primary mechanism for throttling a thread that is executing an instruction that takes a long time to complete, such as a *sync* instruction or a dispatch stall due to a private resource.

Thread priority

Adjustable thread priority allows software to determine when one thread should have a greater (or lesser) share of execution resources. All software layers—operating systems, middleware, and applications—can set the thread priority. Some priority levels are reserved for setting by a privileged instruction only. Reasons for choosing an imbalanced thread priority include the following:

- *A thread is in a spin loop*[5] *waiting for a lock*. Software will give the thread lower priority because it is not doing useful work while spinning.
- *A thread has no immediate work to do and is waiting in an idle loop*. Again, software will give this thread lower priority.
- *One application must run faster than another*. For example, software will give higher priority to a thread running a real-time task than to a thread running a background task.

The POWER5 microprocessor supports eight software-controlled priority levels for each thread. Level 0 is in effect when a thread is not running. Levels 1 (the lowest) through 7 apply to running threads. The POWER5 chip observes the difference in priority levels between the two threads and gives the one with higher priority additional decode cycles. **Figure 4** shows conceptually how the difference in thread priority affects the relative performance of each thread. If both threads are at the lowest running priority (level 1), the microprocessor assumes that neither thread is doing meaningful work and throttles the decode rate to conserve power.

Performance modeling results

As noted earlier, a detailed performance model was used throughout the design to study design alternatives. In this section we present the results from three such studies undertaken to assess the behavior of the POWER5 SMT design. The data presented was obtained from the POWER5 performance model, a trace-based, cycle-accurate model of the processor core, the cache, and the memory subsystem. The data is from a uniprocessor simulation.

The first case shows how the GCT entries are allocated in an SMT environment. **Figure 5(a)** shows a commercial workload from a Java** server application which has moderate cache-miss rates. It achieved a 40% performance gain in SMT mode over ST mode. In SMT mode, an average of 12.7 GCT entries are in use. The threads have identical behavior. In ST mode only 10.9 GCT entries are in use. **Figure 5(b)** shows a mix of two different workloads. Workload0, from a C program of combinatorial optimization, has a high number of cycles per instruction (CPIs) and high cache-miss rates. When run against itself, it achieves a 25% gain over SMT. In general, the higher the CPI, the greater the opportunity for SMT to provide a performance benefit. Cache-miss rates tend to increase with SMT, but the increased thread parallelism more than offsets the higher cache-miss rates [7]. In ST mode, Workload0 uses an average of 14.4 GCT

[5]A thread that fails to obtain a lock often executes instructions in a small loop, called a *spin loop*, checking a flag that indicates whether or not the lock has become available.

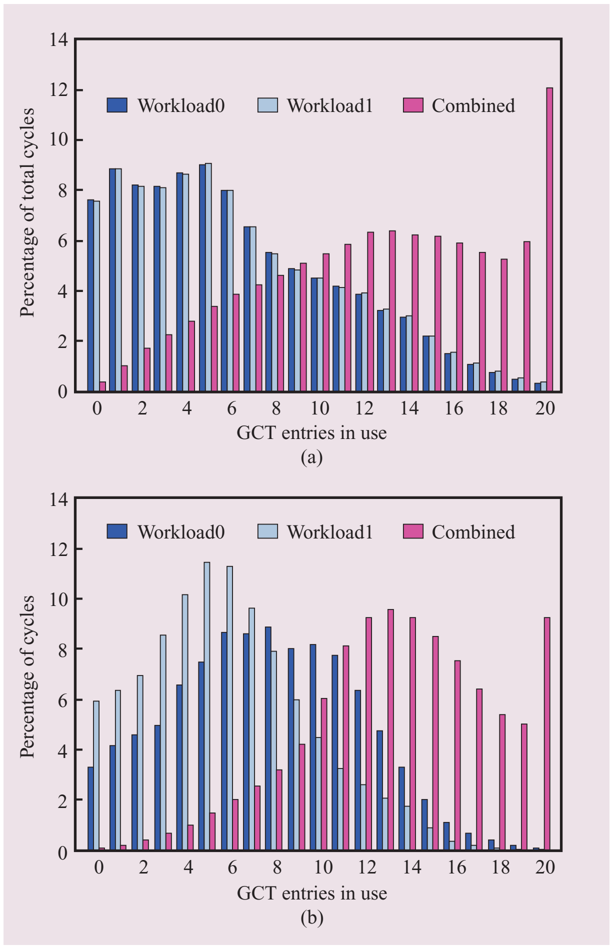

Figure 5

Histogram of GCT entry usage when threads are doing (a) similar workload; (b) different workload.

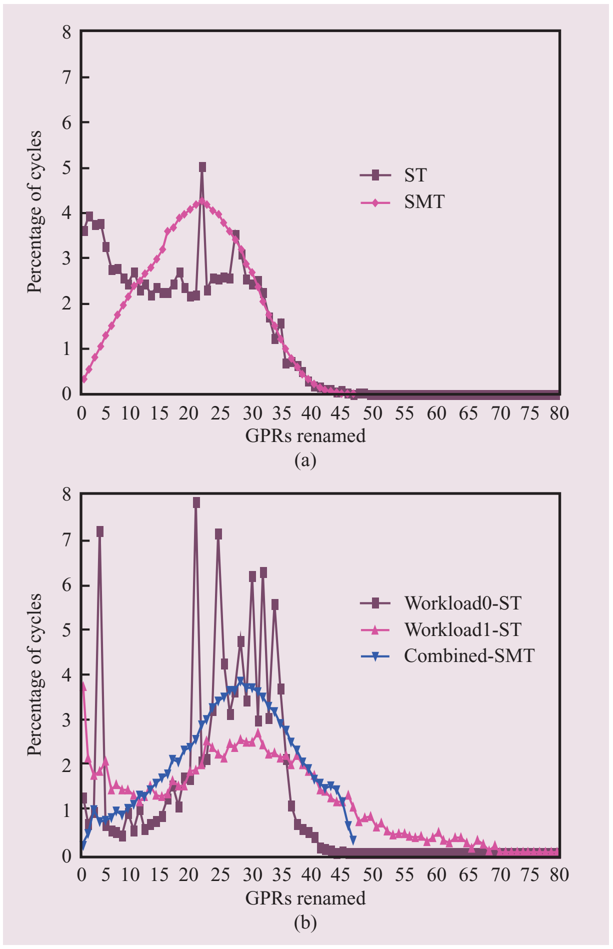

Figure 6

Histogram of GPR registers used above minimum required when threads are doing (a) similar workload; (b) different workload.

entries. Workload1 is from a C program that solves a chessboard layout and has a low CPI and low cache-miss rates. When run against itself, it achieves a 10% SMT gain, consistent with the observation that lower-CPI applications tend to benefit less from SMT. In ST mode, Workload1 uses an average of 11.1 GCT entries. When the workloads are run together, the overall instruction throughput increases by 12% with SMT and uses 13.6 GCT entries. The instruction throughput rate of the individual threads changes by a similar amount for the two threads compared with their single-threaded throughput. In this SMT case, the number of GCT entries used by a thread adapts to the needs of the thread. Thread1 executes instructions at a faster rate and requires fewer GCT entries. The charts also illustrate a basic benefit of SMT. For each of the individual threads, there are frequently no GCT entries in use. This can happen after a branch mispredict or an instruction cache miss. For SMT, it is rare for there to be no GCT entries in use. More frequently than in ST mode, all are in use. This suggests that an increased number of GCT entries would benefit performance. In fact, performance modeling in general has validated this assertion. However, the increased performance is quickly limited by other resources becoming full.

Figure 6 shows a histogram of the number of GPRs renamed for ST mode and SMT mode above the minimum required. The two charts are for the same set of workloads as in Figure 5. In Figure 6(a), the ST case rarely uses more than 50 registers of the maximum 84 available. The SMT case rarely uses the maximum of 48. The shapes of the curves are very similar as they approach the SMT maximum. In Figure 6(b) the two workloads are shown in ST mode as well as SMT mode. Workload1

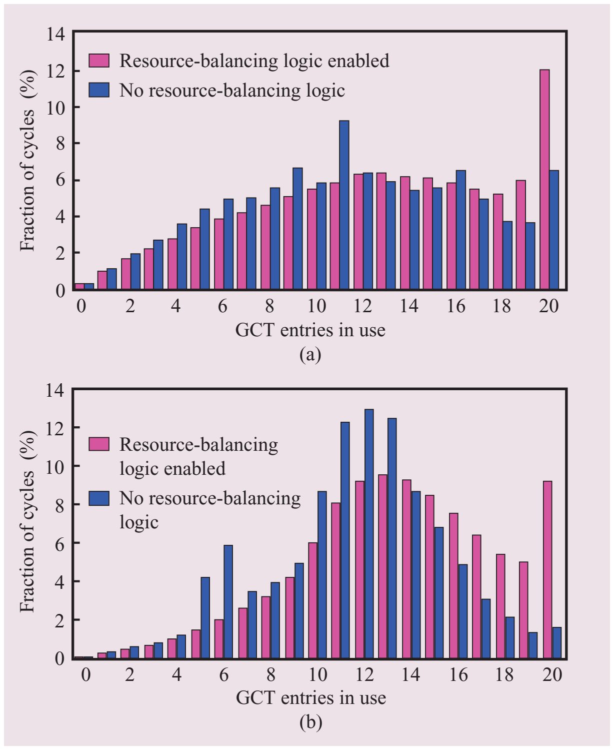

Figure 7

Histogram of GCT entry usage when threads are doing (a) similar work; (b) different work with and without resource-balancing logic.

running in ST mode at times uses more registers than are available in the POWER4 microprocessor. As noted, in ST mode all 120 GPR physical registers are available for use by the thread. The ability to use more registers in ST mode on a POWER5 system than are available in a POWER4 system (80 total, which corresponds to 44 registers being available for renaming) does provide a performance increase. Workload0, running in ST mode, does not use more registers than were available with the POWER4 microprocessor. As shown by the combined-SMT curve in Figure 6(b), when workload0 and workload1 are run together in SMT mode, the number of renames in use is somewhat limited by the total number of rename registers available.

Figure 7 shows the benefit of the resource-balancing logic. The two test cases previously used were run again. However, this time the resource-balancing logic was disabled. The usage histogram with and without the resource-balancing logic is displayed in Figure 7. Resource-balancing logic increases instruction-level parallelism. This is evident from the higher number of GCT entries in use when resource-balancing logic is enabled. The average number of GCT entries increased from 11.5 to 12.7 entries for the two similar workloads [Figure 7(a)], and from 11.5 to 13.6 in the case of the two different workloads executing concurrently in different threads [Figure 7(b)].

Single-threaded operation

Not all applications benefit from SMT. Applications whose performance is execution-unit-limited or which are consuming all of the POWER5 chip memory bandwidth will not see additional performance by having two such threads executing on the same processor. For this reason, POWER5 systems support single-threaded execution mode. In single-threaded mode, a POWER5 system makes all of the rename registers, issue queues, the LRQ, and the SRQ available to the active thread. This gives the single active thread more rename registers to use, allowing it to achieve higher performance levels than a POWER4 system at equivalent frequencies. Software can dynamically change a processor between single-threaded and SMT modes.

The POWER5 chip is designed to be booted initially in single-threaded mode. System firmware can then enable multithreading in three ways:

1. A move to a special-purpose register that controls threading can be executed to wake up the inactive thread.
2. Firmware can set up the inactive thread to wake up on an external or decrementer interrupt presented to the inactive thread.
3. The service processor can present a system reset interrupt to the inactive thread.

Once a thread is running, the only way to shut it down is by software performing a move to a special-purpose register to stop it. When this is executed, all resources used by that thread are released to the active thread. If the thread is subsequently activated, software must restore the architected state. Hence, it is necessary for software to save the architected state of the thread before stopping it.

If the machine is running in single-threaded mode and software attempts to stop the only remaining thread, the operation is ignored.

Memory subsystem

The POWER5 memory controller provides the address and controls to support a data port to main store. It uses a synchronous memory interface (SMI) to DDR-I or DDR-II SDRAM. The support is provided for connection to either two or four synchronous memory interface chips located near the memory DIMMs. Up to two DIMMs behind each of two ports can be attached to each SMI chip.[6] In the two-SMI mode, each SMI chip is

[6]The actual number of DIMMs per SMI chip and SMI chips per system is system-dependent.

configured for an 8-byte read and 2-byte write interface on the controller side, and two independent 8-byte ports to the DIMMs. In the four-SMI mode, each SMI chip is configured for a 4-byte read and a 2-byte write interface on the controller side and two independent 8-byte ports to the DIMMs. The SMI chips contain buffers to match the differences in interface widths between the controller side and the DIMMs.

The memory controller logic straddles two clock domains which operate asynchronously to each other. The clock speed of the memory clock domain is determined by the speed of the DRAM modules to be used in the system. It operates at twice the DRAM module frequency. Memory is protected by a single-bit error correct, double-bit error detect error-correction code (ECC). Additionally, memory scrubbing[7] is used in the background to find and correct soft errors. Each memory extent has an extra DRAM to allow for transparent replacement of one failing DRAM per group of four DIMMs using chipkill technology.[8]

A logical view of the memory subsystem is shown in **Figure 8**. The interface between the POWER5 chip and the SMI chips is made up of three buses. All of these buses operate at twice the DIMM speed. The three buses are the address/command bus, the unidirectional write data bus, and the unidirectional read data bus. The write data bus is 8 bytes wide, with each of up to four SMI chips receiving 2 bytes point to point. In configurations with two SMI chips per POWER5 chip, four of the 8-byte write data bus pins are left unconnected. The read data bus consists of 16 bytes. When four SMI chips are used, each SMI drives four of the 16-byte read data bus inputs. In configurations with two SMI chips per POWER5 chip, each SMI chip drives eight of the 16-byte read data bus inputs.

Interconnecting chips to form larger SMPs

Two basic building blocks are used to build a POWER5 system: a multichip module (MCM) with four POWER5 and four L3 chips and a dual-chip module (DCM) with one POWER5 chip and one L3 chip. Depending on the system, two to eight MCMs or one to eight DCMs are packaged together in a system. Information flow between POWER5 chips in a system is managed in a distributed manner by a fabric bus controller (FBC) located on each POWER5 chip. This interconnect topology, referred to as a distributed switch, functions in a similar manner across all configurations.

[7]To prevent single-bit errors adding up into noncorrectable multibit errors, the memory controller employs a memory-scrubbing process. In this process, the memory controller reads memory locations during idle periods, detects single-bit errors, and writes the corrected data back to the memory.

[8]The use of IBM chipkill technology [8] allows detection and correction of most multibit memory errors. Since chipkill technology distributes information covered by error-correction coding across separate memory chips, if any of the chips fail, the data can still be reconstructed from the remaining chips and the system can continue running.

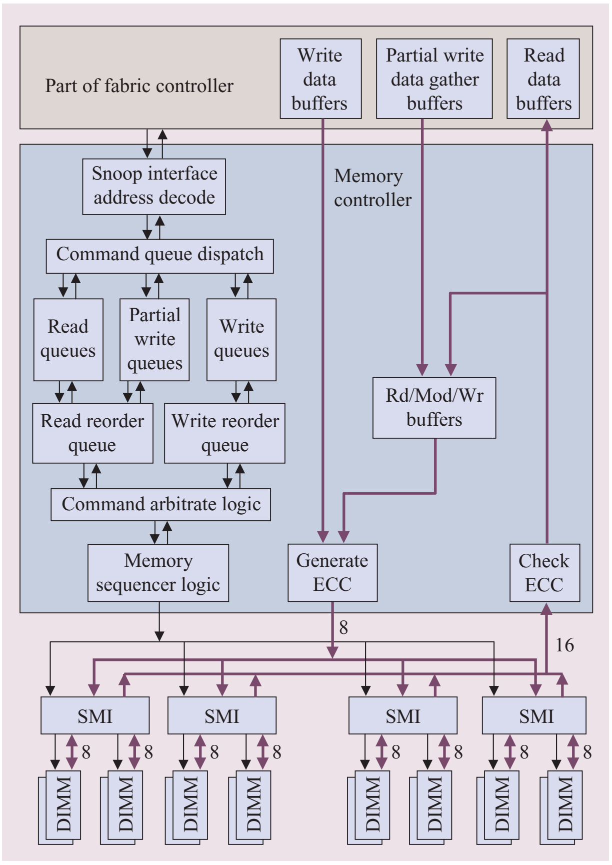

Figure 8

POWER5 memory subsystem logical view.

Systems based on multichip modules

High-end POWER5 systems use the MCM package. The basic building block is eight POWER5 chips packaged with eight L3 chips on two MCMs. The MCMs, associated memory, and bridge chips to connect to I/O drawers comprise a book. Interconnection among the POWER5 chips, L3 chips, and memory is shown in **Figure 9**.

Connections from each POWER5 chip to L3, memory, and I/O are made through separate pairs of dedicated unidirectional buses. The buses to the L3 cache, 16 bytes wide in each direction, operate at half the processor frequency and scale with the processor frequency. The I/O bus, 4 bytes wide in each direction, referred to as the GX bus, operates at one third of the processor frequency. The memory bus operates at twice the DRAM frequency and scales with DRAM frequency. POWER5 chips on the MCM are interconnected with two buses in a ring configuration, with data flowing in opposite directions on the two buses. Each of these buses is 8 bytes wide and operates at processor frequency.

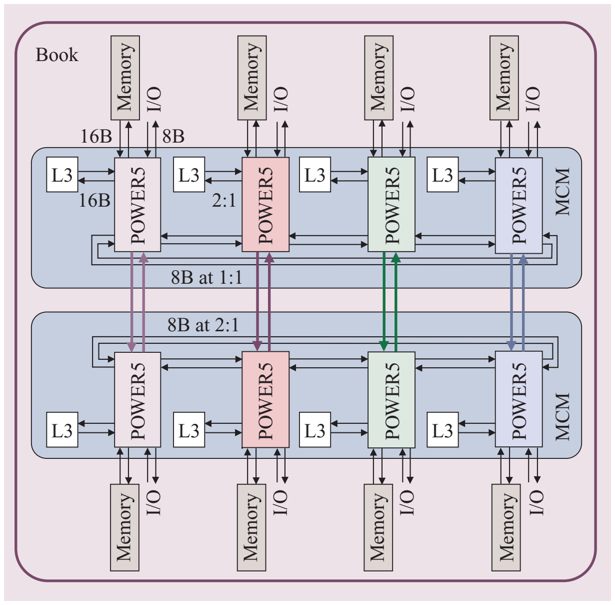

Figure 9

POWER5 16-way building block.

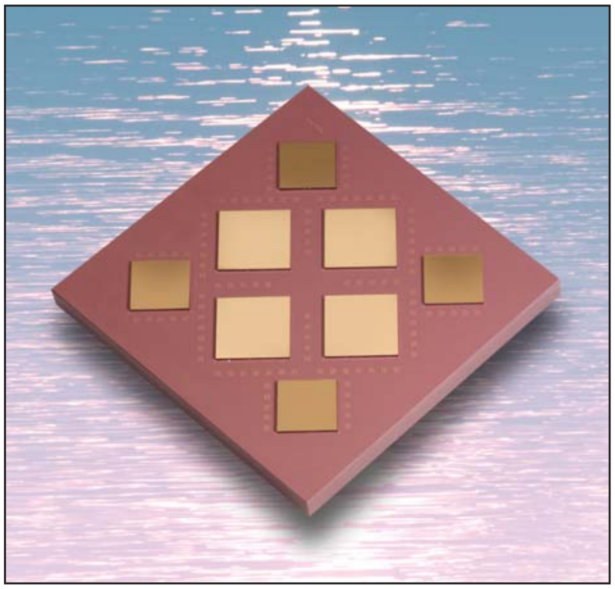

Figure 10

POWER5 multichip module.

The POWER5 MCM, shown in **Figure 10**, is a high-performance ceramic substrate with 89 levels of metal measuring 95 mm on a side. The module itself is strictly passive. The MCM configuration is replicated with an identical module on a processor board to form a book. Four pairs of unidirectional data buses, each 8 bytes wide and operating at half processor frequency, interconnect the POWER5 chips on the two modules. A POWER5 book is a 16-way symmetric multiprocessor; with simultaneous multithreading, this presents a 32-way image to software.

To build systems larger than 16-way, multiple books are interconnected as shown in **Figure 11(a)**. Up to four books can be interconnected to build up to a 64-way symmetric multiprocessor. POWER5 chips in each book are connected to corresponding chips in adjacent books, as shown with an 8-byte-wide bus operating at half processor frequency.

Systems based on dual-chip modules

An alternate configuration based on a dual-chip module is used to form systems with one to eight POWER5 chips. DCMs are interconnected similarly to MCMs, with the exception that there is only one POWER5 chip and its associated L3 chip. A 16-way DCM-based POWER5 system is shown in **Figure 11(b)**. Two DCMs serve as the basic building blocks. Systems can be constructed with one, two, four, six, or eight DCMs. **Figure 12** is a photograph of the POWER5 DCM.

Fabric bus controller

A primary element enhancing POWER5 system scalability is the flexibility and scalability of the fabric bus controller on each POWER5 chip. The FBC buffers and sequences operations among the L2/L3, the functional units of the memory subsystem, the fabric buses that interconnect POWER5 chips on the MCM, and the fabric buses that interconnect multiple MCMs.[9] The fabric bus has separate address and data buses that run independently to allow split transactions. Transactions are tagged to allow out-of-order replies.

Address bus description

The inter-MCM address buses are 8 bytes wide, operating at half the processor frequency, while the intra-MCM address buses are 4 bytes wide, operating at full processor frequency. Every address transfer uses the bus for four processor cycles. Address bus operations are protected by ECC that provides for single-error correction and double-error detection (SECDED). Address transfers include a 50-bit real address, an address tag, transfer type, and other relevant data that uniquely identifies each address as it appears on the bus. There are twelve point-to-point address buses interconnecting the four POWER5 chips on an MCM.

The fabric broadcasts addresses from MCM to MCM using a ring structure. The address propagates serially

[9]MCM configurations are used to describe the fabric operation. In systems built with DCMs the operation is identical, with the exception that there are no intra-module POWER5-to-POWER5 chip connections.

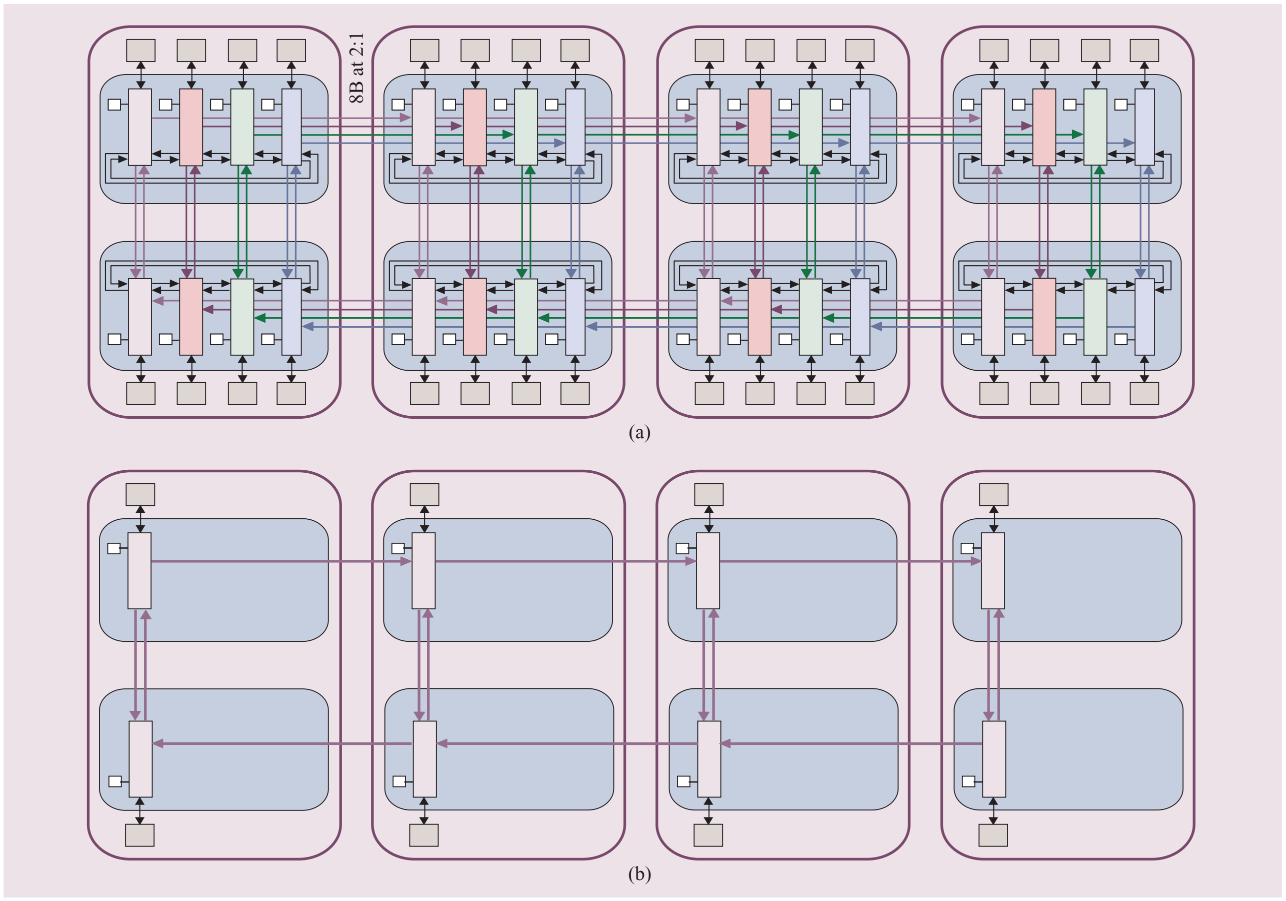

Figure 11

(a) POWER5 64-way logical structure. (b) POWER5 16-way system built with eight dual-chip modules.

through all eight MCMs in a 64-processor system. Each chip on the ring that initiates the address or receives it from another module is responsible for forwarding the address to the next MCM in the ring and to the other chips on its own MCM. Once the originating chip receives the address it transmitted, it does not continue propagating it.

Response bus description

Similarly to the address buses, the inter-MCM response buses run at half the processor frequency. The intra-MCM response buses operate at the processor frequency. The response bus is essentially a delayed version of the address bus. It includes information related to the cache coherency actions that must be taken after the memory subsystem units on each processor chip have snooped[10] the address against its corresponding queues and directories. The response bus phase is parity-protected. It follows the address phase delayed by a fixed number of cycles on the same set of buses; i.e., once a chip snoops an address, the chip must return a snoop response to the source chip or the chip on the source ring in a fixed number of cycles, with the actual number of cycles dependent on system size. Once the source chip on the MCM receives the snoop responses from the other three chips on the MCM, it combines these responses with its own response and with the collective response obtained from the previous MCM in the ring and forwards the merged snoop response to the next MCM in the ring for further accumulation.

When the originating chip receives the snoop response from the system for the address it initiated, it generates a combined response that is broadcast throughout the system just like an address phase. The combined response details the ultimate action to be taken on the corresponding address (for example, what state the master can now go to, which chip is to supply the data to the master for a read operation, and any

[10]When an address is obtained by a processor chip from the global address bus, the process is known as *snooping*.

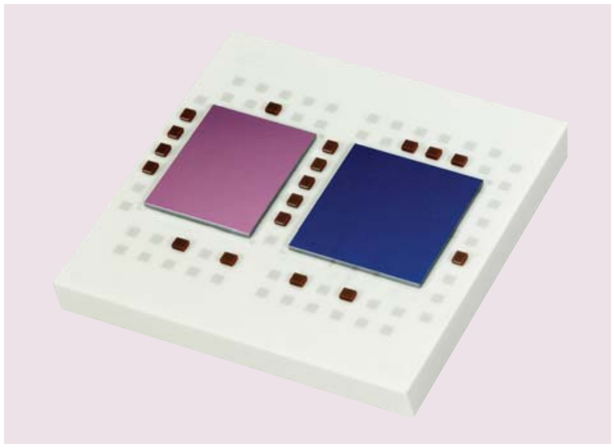

Figure 12

POWER5 dual-chip module.

invalidate scenarios that must be performed in the background).

To reduce cache-to-cache transfer latency, POWER5 systems add the notion of an early combined response mechanism, which allows a remote chip to send a line from its L2 (or L3 as the case may be) in shared state to the requesting processor soon after it receives the address. The early combined response is determined by comparing an MCM snoop response with the cumulative snoop response of all previous MCMs in the ring, in order to find the first coherent snooper that can supply the read data. This early combined response mechanism allows the initiation of the data-transfer phase by a coherent snooper soon after it receives the address. When such a coherent snooper is found, the early combined snoop response sent to the downstream snoopers also indicates that the source of data has already been found.

Data bus description

The four POWER5 chips on an MCM are interconnected by two unidirectional rings transmitting data in opposite directions. Like the address bus, the data bus on the MCM also runs at the processor frequency and scales with it.

The processor chips on the two MCMs in a book are connected by vertical data buses as shown in Figure 9. The four processor chips on an MCM that belong to different books are connected by four separate rings as shown in Figure 11(a).

The data bus services all data-only transfers, such as cache interventions. It also services the data portion of address-initiated operations that require data to be sent, such as cast-outs[11], snoop pushes[12], and DMA writes.

The data bus phase, which is ECC (SECDED)-protected, has been enhanced for POWER5 systems. In POWER4 systems, the data propagated in a simple ring structure and followed a broadcast model on the destination MCM. In the POWER5 design, the number of data buses within a module has been doubled to eight. Between modules the data buses have also been doubled to eight buses. In addition, in POWER5, the fabric routes data to the specific chip that requested it. Also, the vertical-node data buses, in addition to the traditional node-to-node data buses, were added to the system for POWER5 to provide a shorter path with reduced latency.

Data prefetching

Like POWER4 systems, POWER5 systems employ hardware to prefetch data into the L1 data cache. When load instructions miss sequential cache lines, either ascending or descending, the prefetch engine initiates accesses to the following cache lines before being referenced by future load instructions. In order to ensure that the data will be in the L1 data cache when needed, the L1 data cache prefetch is initiated when a load instruction references data from a new cache line. At the same time, the transfer of a line into L2 from memory is requested. Since the latency for retrieving a line of data from memory into the L2 is longer than that for moving it from L2 to the L1, the prefetch engine requests data from memory twelve lines ahead of the line being referenced by the load instruction. Eight such streams per processor are supported.

To guard against premature initiation of a data-prefetch stream by the hardware, the prefetch engine ramps up the prefetches slowly, requiring an additional two sequential cache misses to occur before reaching the steady-state prefetch sequencing. In POWER4 systems, the data cache block touch (*dcbt*) instruction was extended to indicate to the hardware that a prefetch stream should be installed immediately, without waiting for confirmation. This software-initiated data prefetching continues to be supported on POWER5 systems and is enhanced to indicate to the hardware the extent of the prefetching, i.e., how many lines to prefetch. When the pre-specified number of lines has been prefetched, using the same mechanism as with hardware-initiated prefetching, the stream is terminated. Software-directed prefetching has two benefits: It improves the performance for short streams by eliminating the initial ramp-up, and

[11]When a cache line is replaced, the original data is *cast out* (that is, written back to a lower-level cache or to the memory). If the data is not modified, it may simply be overwritten, without any cast-out.

[12]When a cache line is transferred from one processor to another processor because of a snoop response, the action is referred to as a *snoop push*.

it eliminates prefetching unnecessary lines by specifying the last line to prefetch.

Dynamic power management

In current CMOS technologies, chip power is an important design parameter from both the consumption and the cooling perspectives [9]. In addition to the technology challenges associated with leakage power, switching power has also increased because of the increase in core area and the higher instruction-execution rates made possible with SMT. Additionally, the many buses implemented per POWER5 chip have increased the switching power used in driving off chip to other chips. This has resulted in a net increase in both total power and power density for the POWER5 chip over the POWER4+ chip operating at the same frequencies. Note that both are implemented in the same technology family.

To reduce switching power, the POWER5 design makes extensive use of a fine-grain, dynamic clock-gating mechanism. Fine-grain clock-gating allows a local clock buffer to be gated off if the set of latches it drives is known not to be written in the next cycle. This allows substantial power saving with no impact on performance.

In every cycle, POWER5 dynamic power management logic determines whether or not a local clock buffer (LCB) has to be clock-gated in the next cycle. When an LCB is clock-gated, the set of latches it drives can still be read, but they cannot be written. Power-modeling tools were used to estimate the utilization of various design macros and their associated switching power across a range of workloads and then determine the benefit of clock-gating all or part of those design macros. Cycle-by-cycle dynamic power management was implemented in those macros, providing a reasonable power-saving benefit. Special attention was paid to ensure that there is no performance loss due to clock-gating, and that clock-gating logic does not create a critical timing path. Logic to determine a clock-gating event was kept simple to minimize overhead.

To reduce leakage power, there is minimal use of transistors with low threshold voltage in the POWER5 chip. High-threshold-voltage transistors are used in most arrays and other key places in the chip.

As noted in the discussion on thread priority, the POWER5 chip operates in low-power mode when software instructs the hardware to execute both threads at the lowest available priority. In low-power mode, instructions are dispatched at most once every 32 cycles. By convention, this is used only when there is no ready software task to run on either thread on a processor core, further reducing switching power. **Figure 13** shows the effects of dynamic power management with and without SMT enabled.

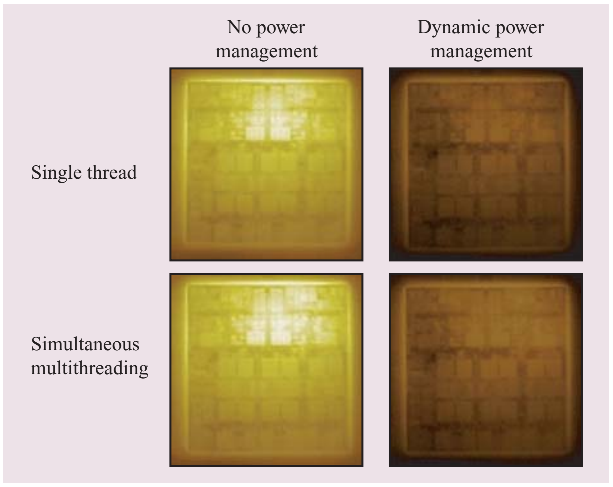

Figure 13

Effects of dynamic power management with and without simultaneous multithreading enabled. Photographs were taken with a heat-sensitive camera while a prototype POWER5 chip was undergoing tests in the laboratory.

To protect against adverse environmental factors, the POWER5 chip has 24 strategically located digital temperature sensors [10]. When an over-temperature condition occurs, the sensors signal the core control logic to engage a two-stage temperature-reducing response. The first stage reduces average switching and clocking power by rapidly alternating between execution and stall conditions. Once the temperature falls below a reset value, normal operation resumes. If the first thermal response fails to reduce the temperature within an acceptable time period, the second stage engages and dramatically reduces temperature by prolonged throttling of processor throughput via functions such as fetch, dispatch, or completion. This throttling response requires no software or service processor intervention. It provides timely and flexible protection for one-way to 64-way systems while minimizing performance impact.

Reliability, availability, and serviceability characteristics

POWER4 systems were designed to provide a high level of availability and data integrity in the presence of hardware failures [11]. High availability was achieved by minimizing component failure rates through improvements in the base technology, and through design techniques that permit hard- and soft-failure detection, recovery, and isolation, repair deferral, and component replacement concurrent with system operation. Fault-tolerant design techniques were used for array, logic, storage, and I/O subsystems. A diagnostic strategy that included fault-isolation and recovery techniques was employed. With POWER4 systems, our philosophy was to minimize any form of outage, reducing the need for redundant components [11].

With POWER5 systems, this strategy continues, with new emphasis on reducing scheduled outages to further improve system availability. On POWER5 systems, many firmware upgrades can be done on a running machine without any reboot or impact to the system. ECC has been implemented on all system interconnects. Single-bit interconnect failures are dynamically corrected. If the failure is persistent, a deferred repair action can be scheduled. As with POWER4 systems, first failure data capture capabilities are used to identify the source of a nonrecoverable failure. When a nonrecoverable error is encountered, the system is taken down, the book containing the fault is taken offline, and the system rebooted, all without human intervention. Thermal protection sensors, described in the power-management section of this paper, are used to guard against environmental factors that could cause an over-temperature condition that would otherwise be injurious to the system.

Summary

POWER5 systems are designed to allow higher levels of performance by allowing higher levels of symmetric multiprocessing, up to 64 real processors, and higher per-processor performance than POWER4 systems. Processor performance is increased by allowing each processor to be two-way threaded using SMT. Additionally, single-threaded performance has been enhanced. POWER5 systems can operate in either single-threaded or simultaneous multithreaded modes. Binary compatibility has been maintained with POWER4 systems. By maintaining the same pipeline structure in both systems, optimizations performed for POWER4 systems work on POWER5 systems. To conserve power, the POWER5 design and implementation has introduced a unique capability that permits power savings without affecting performance.

Acknowledgments

The effort of designing, developing, verifying, and testing any new computer system involves numerous individuals. In our case, this included dedicated engineers and programmers from several IBM research and development laboratories. Since the list of contributors to be recognized is too lengthy to include here, we as authors are honored to document the contribution of a large, proud team.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Sun Microsystems, Inc.

References

1. J. M. Tendler, J. S. Dodson, J. S. Fields, Jr., H. Le, and B. Sinharoy, "POWER4 System Microarchitecture," *IBM J. Res. & Dev.* **46**, No. 1, 5–25 (January 2002).
2. R. Kalla, B. Sinharoy, and J. M. Tendler, "IBM POWER5 Chip: A Dual-Core Multithreaded Processor," *IEEE Micro* **24**, No. 2, 40–47 (March/April 2004).
3. J. M. Borkenhagen, R. J. Eickmeyer, R. N. Kalla, and S. R. Kunkel, "A Multithreaded PowerPC Processor for Commercial Servers," *IBM J. Res. & Dev.* **44**, No. 6, 885–898 (November 2000).
4. R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Porterfield, and B. Smith, "The Tera Computer System," presented at the 1990 ACM International Conference on Supercomputing, Amsterdam, Netherlands, June 1990.
5. D. M. Tullsen, S. J. Eggers, and H. M. Levy, "Simultaneous Multithreading: Maximizing On-Chip Parallelism," *Proceedings of the 22nd Annual International Symposium on Computer Architecture,* June 1995, pp. 392–403.
6. C. May, E. Silha, R. Simpson, and H. Warren, *The PowerPC Architecture,* Morgan Kaufmann Publishers, San Francisco, 1994.
7. O. Herescu, B. Olszewski, and H. Hua, "Performance Workloads Characterization on POWER5 with Simultaneous Multi Threading Support," *Proceedings of the Eighth Workshop on Computer Architecture Evaluation Using Commercial Workloads,* February 2005, pp. 15–23.
8. See *http://www-1.ibm.com/servers/eserver/pseries/campaigns/chipkill.pdf*.
9. D. Brooks, P. Bose, V. Srinivasan, M. K. Gschwind, P. G. Emma, and M. G. Rosenfield, "New Methodology for Early-Stage, Microarchitecture-Level Power–Performance Analysis of Microprocessors," *IBM J. Res. & Dev.* **47**, No. 5/6, 653–670 (September/November 2003).
10. J. Clabes, J. Friedrich, M. Sweet, J. DiLullo, S. Chu, D. Plass, J. Dawson, P. Muench, L. Powell, M. Floyd, B. Sinharoy, M. Lee, M. Goulet, J. Wagoner, N. Schwartz, S. Runyon, G. Gorman, P. Restle, R. Kalla, J. McGill, and S. Dodson, "Design and Implementation of the POWER5 Microprocessor," *ISSCC Digest of Technical Papers,* February 2004, pp. 56–57.
11. D. C. Bossen, A. Kitamorn, K. F. Reick, and M. S. Floyd, "Fault-Tolerant Design of the IBM pSeries 690 System Using POWER4 Processor Technology," *IBM J. Res. & Dev.* **46**, No. 1, 77–86 (January 2002).

Received March 2, 2005; accepted for publication June 27, 2005; Internet publication September 7, 2005

Balaram Sinharoy *IBM Systems and Technology Group, 2455 South Road, Poughkeepsie, New York 12601 (balaram@us.ibm.com).* Dr. Sinharoy is a Distinguished Engineer in the IBM Systems and Technology Group. He has been the Chief Scientist for the POWER5 microprocessor design, responsible for the POWER5 microarchitecture definition. Prior to his work on the POWER5 microprocessor, he led many architecture, microarchitecture, performance and verification efforts for the POWER4 microprocessor. His research and development interests include advanced microprocessor design, computer architecture, and performance analysis. Dr. Sinharoy has published numerous articles and received numerous patents in these areas. He received an IBM Outstanding Technical Achievement Award, an IBM Outstanding Innovation Achievement Award, and an IBM Corporate Award for his contributions to POWER4 and POWER5 microprocessors, as well as an IBM Fourteenth Plateau Invention Achievement Award and an IBM Fourth Plateau Publication Achievement Award. Dr. Sinharoy is a Senior Member of IEEE and has been named an IBM Master Inventor.

Ronald N. Kalla *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (rkalla@us.ibm.com).* Mr. Kalla is the lead engineer for IBM POWER5, specializing in processor core development. He has designed processor cores for S/370*, M68000, AS/400*, and RS/6000* machines. He received an IBM Outstanding Innovation Award and an IBM Corporate Award for his contributions to the POWER5 system. Mr. Kalla holds numerous patents on processor architecture. He also has an extensive background in post-silicon hardware bring-up and verification. He has 12 issued U.S. patents, with 15 additional patents pending, and has published 15 technical disclosures.

Joel M. Tendler *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (jtendler@us.ibm.com).* Dr. Tendler is program director of technology assessment, responsible for assessing emerging technologies for applicability in future eServer* iSeries* and pSeries* product offerings. He has extensive hardware and software design experience in S/390* and RS/6000 systems. Most recently, Dr. Tendler received an IBM Outstanding Technical Achievement Award for his contribution to POWER4 and POWER5 systems. He has several U.S. patents issued and has received an IBM First Plateau Invention Achievement Award.

Richard J. Eickemeyer *IBM Systems and Technology Group, 3605 Highway 52 N., Rochester, Minnesota 55901 (eick@us.ibm.com).* Dr. Eickemeyer is a Senior Technical Staff Member in the IBM Systems and Technology Group. He is currently the processor core performance team leader for the IBM PowerPC servers. Prior to this, he worked on performance and architecture of several processors used in AS/400 systems and S/390 systems in Rochester, Minnesota, and Endicott, New York. Since joining IBM, he has received awards including a Ninth Plateau IBM Invention Achievement Award, an IBM Outstanding Technical Achievement Award, two IBM Outstanding Innovation Awards, and IBM Corporate Awards. He has also been named an IBM Master Inventor. 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. His research interests are computer architecture and performance analysis. He is a Senior Member of the IEEE.

Jody B. Joyner *IBM Systems and Technology Group, 11400 Burnet Road, Austin, Texas 78758 (joyner@us.ibm.com).* Mr. Joyner is a Senior Engineer in the IBM Systems and Technology Group in Austin, Texas. He received a B.S.E.E. degree (1993) and an M.S.E.E. degree (1999) from the University of Texas at Austin. He has specialized in the logic design of high-performance system bus interconnects in the POWER4 and POWER5 programs, along with their follow-ons. He led the hardware bring-up and validation efforts for the POWER5 storage subsystem. Mr. Joyner recently received an IBM Outstanding Innovation Achievement Award and an IBM Corporate Award for his contribution to the POWER5 microprocessor. He was named an IBM Master Inventor in 2000 and has received an IBM Twelfth Plateau Invention Achievement Award.