# 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
P. Williams

*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.*

## 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



**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 $mm^2$ 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).

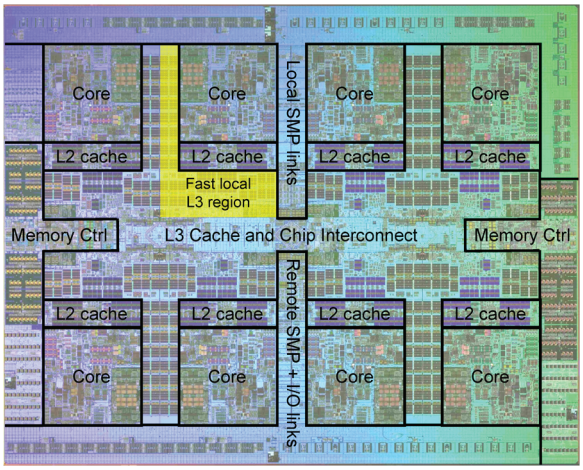


Figure 1

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

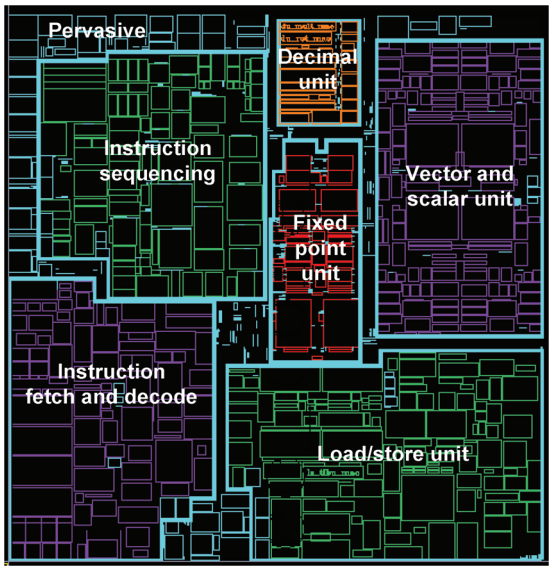


Figure 2

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.

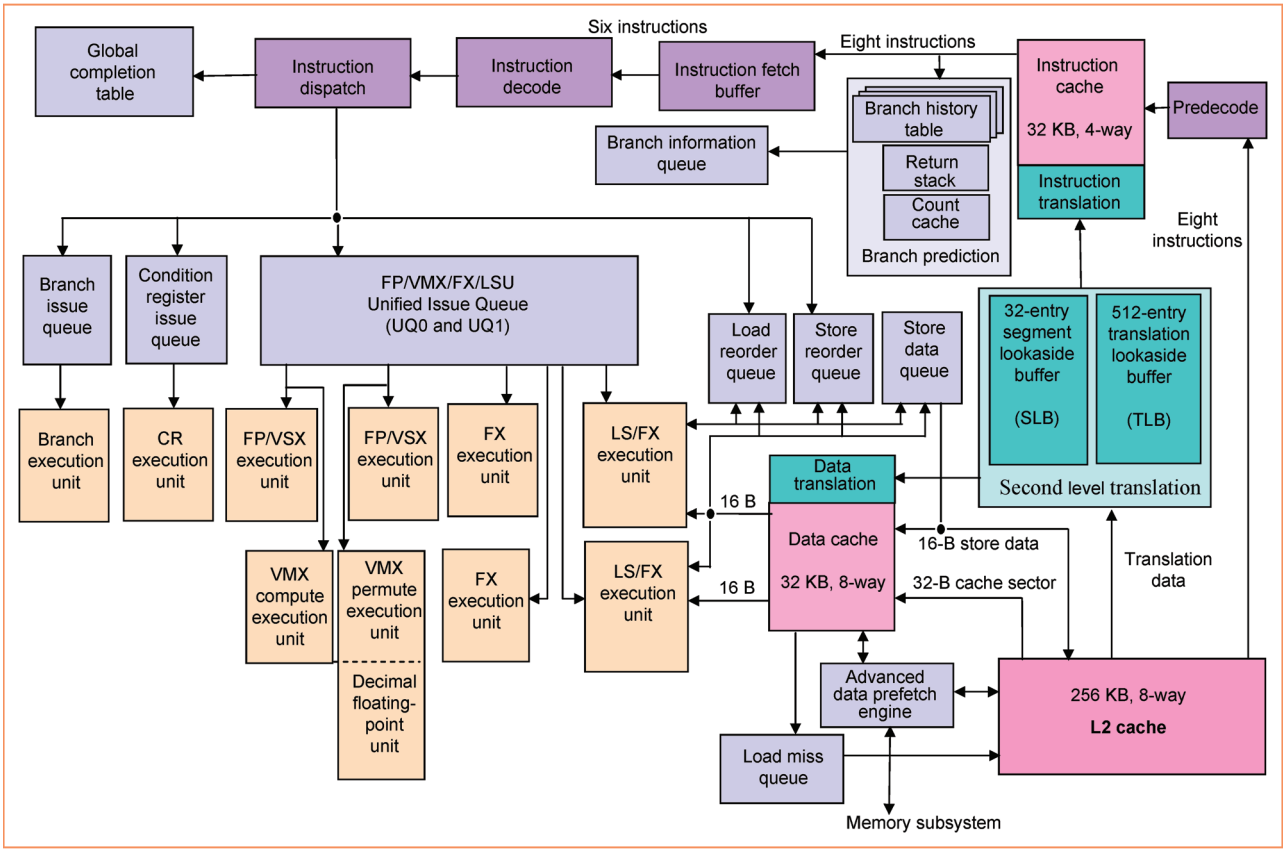


Figure 3

IBM POWER7 processor core pipeline flow.

Each POWER7 chiplet (i.e., a POWER7 core with its Level 2 (L2) and local L3 cache) is designed to be on a separate power domain, with asynchronous boundary with the PowerBus to which it is connected. This allows each chiplet to have independent voltage and frequency slewing for advanced power management.

**Figure 4** shows how instructions flow to the various issue queues and are then sent to the functional units for execution. Which copy of the issue queue, physical register file, and functional unit will be used by an operation depends on the multithreading mode of the processor core. In the ST and SMT2 modes, the two physical copies of the GPR have identical contents. Instructions from the thread(s) can be dispatched to either one of the issue queue halves (UQ0 or UQ1) in these modes. Load balance across the two issue queue halves is maintained by dispatching alternate instructions of a given type from a given thread to a UQ half.

In an SMT4 mode, the two copies of the GPR have different contents. FX and load/store (LS) operations from threads T0 and T1 can only be placed in UQ0, can only access GPR0, and can only be issued to FX0 and LS0 pipelines. FX and LS operations from threads T2 and T3 can only be placed in UQ1, can only access GPR1, and can only be issued to FX1 and LS1 pipelines.

As shown in Figure 4, most VSU operations can be dispatched to either UQ0 or UQ1 in all modes (single thread, SMT2, SMT4), with the following exceptions: 1) VMX floating point and simple and complex integer operations can only be dispatched to UQ0; 2) permute (PM), decimal floating point, and 128-bit store operations can only be dispatched to UQ1; 3) VSU operations dispatched to UQ0 always execute on vector scalar pipeline 0 (VS0); and 4) VSU operations dispatched to UQ1 always execute on VS1 pipeline.

The core consists primarily of the following six units: instruction fetch unit (IFU), instruction-sequencing unit (ISU), LSU, FXU, VSU, and decimal FPU (DFU). The IFU contains a 32-KB instruction cache (I-cache), and the LSU contains a 32-KB data cache (D-cache), which are each backed up by a tightly integrated 256-KB unified L2 cache.

In a given cycle, the core can fetch up to eight instructions, decode and dispatch up to six instructions, and issue and execute up to eight instructions. There are 12 execution units within the core, i.e., two fixed point, two LS,

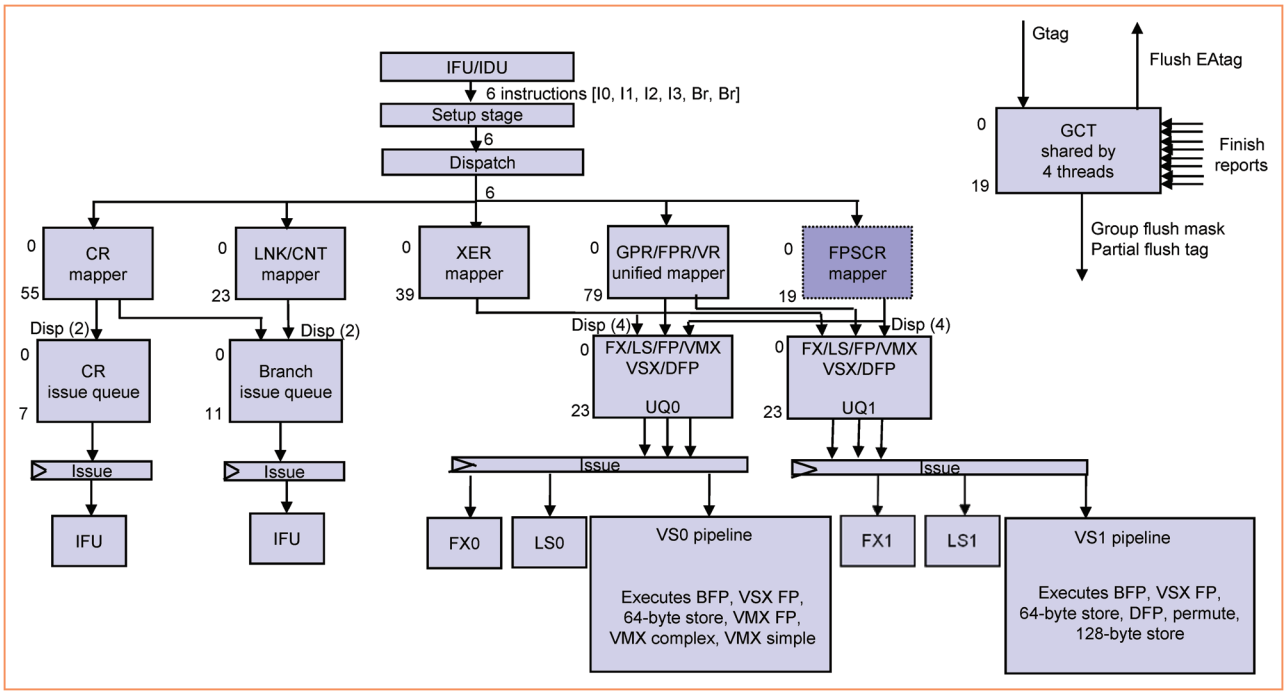


Figure 4

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

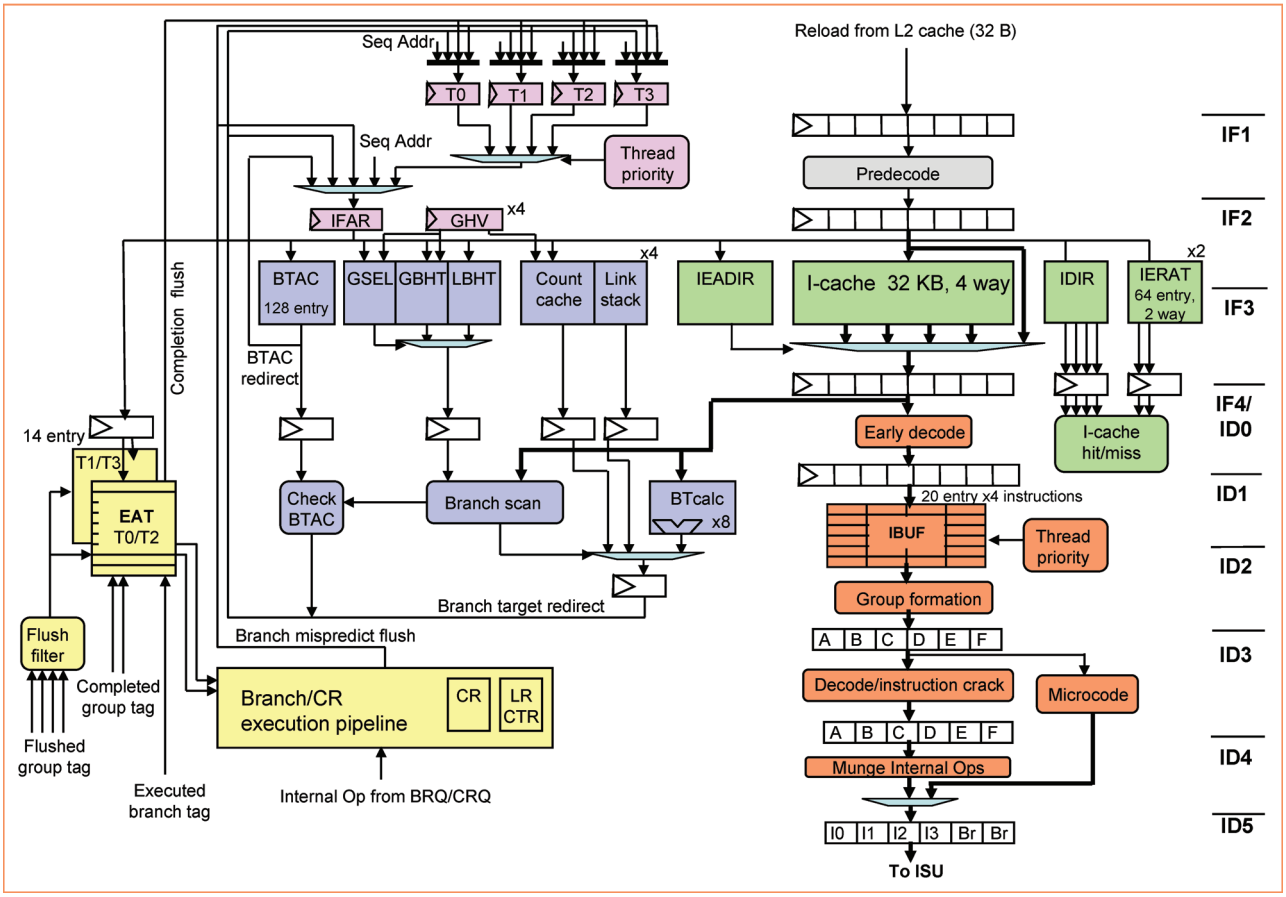


Figure 5

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.

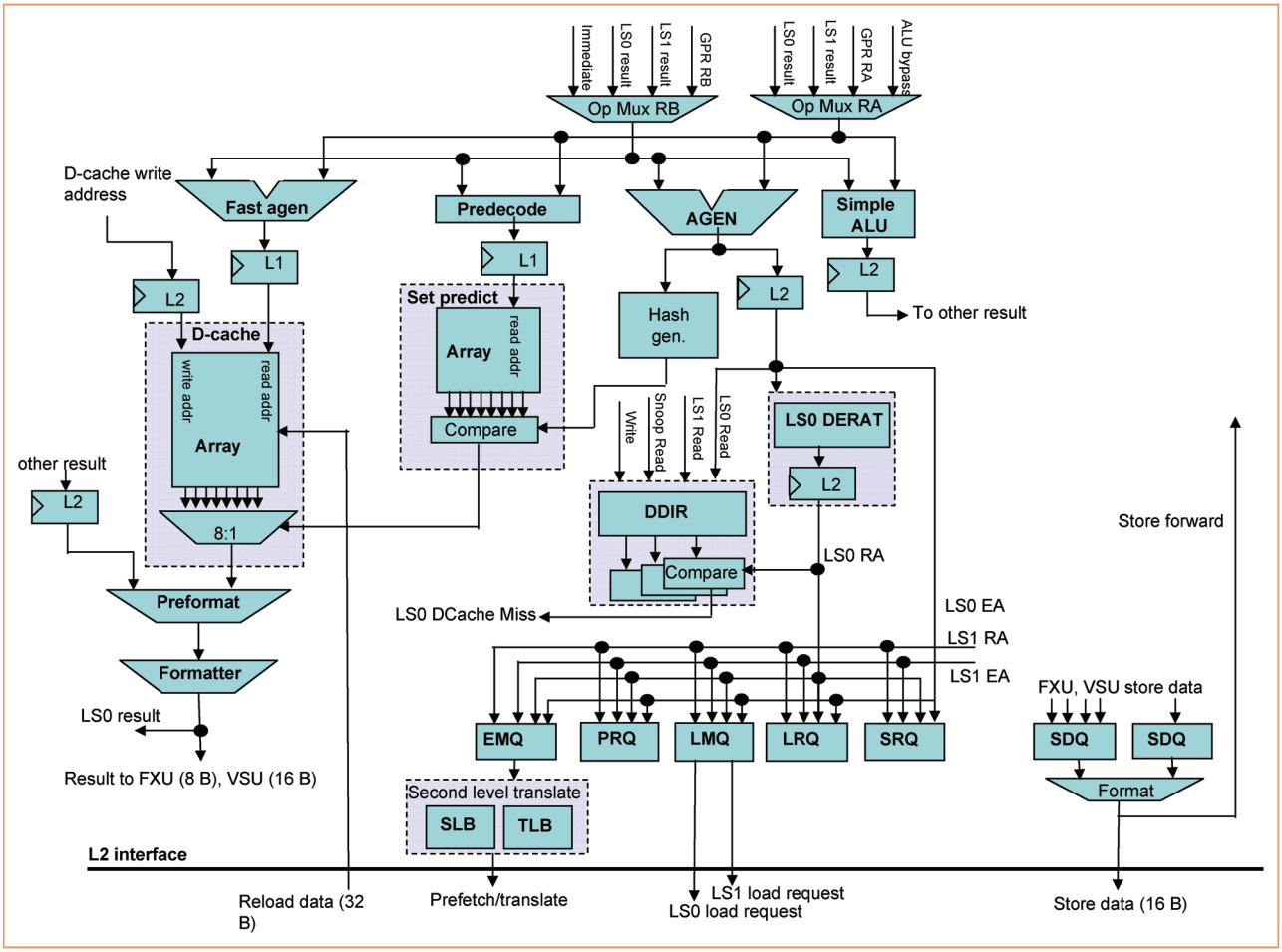


Figure 6

LSU microarchitecture (LS0 pipe shown).

The SRQ is a 32-entry real-address-based content-addressable memory (CAM) structure. Each thread has 64 virtual entries that are available, allowing 64 outstanding stores to be dispatched per thread. A total of 32 outstanding stores may be issued since a real physical SRQ entry is required for the store to be issued. The SRQ is dynamically shared among the active threads. An SRQ entry is allocated at issue time and deallocated after the completion point when the store is written to the L1 D-cache or sent to the L2 cache. For each SRQ entry, there is a corresponding SDQ entry of 16 bytes. Up to 16 bytes of data for a store instruction can be sent to the L2 cache (and also written to the L1 D-cache on a hit) in every processor cycle. Store forwarding is supported, where data from an SRQ entry is forwarded to an inclusive subsequent load, even if the store and load instructions are speculative.

Like the SRQ, the LRQ is a 32-entry real-address-based CAM structure. Sixty-four virtual entries per thread are available to allow a total of 64 outstanding loads to be dispatched per thread. A total of 32 outstanding loads may be issued since a real physical LRQ entry is required for the load to be issued. The LRQ is dynamically shared among the threads. The LRQ keeps track of out-of-order loads, watching for hazards. Hazards generally exist when a younger load instruction executes out of order before an older load or store instruction to the same address (in part or in whole). When such a hazard is detected, if specific conditions exist, the LRQ initiates a flush of the younger load instruction and all its subsequent instructions from the thread, without having an impact on the instructions from other threads. The load is then refetched from the I-cache and reexecuted, ensuring proper LS ordering.

### *Address translation*

In the PowerPC architecture, programs are written using 64-bit effective addresses (EAs) (32-bit in 32-bit addressing

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

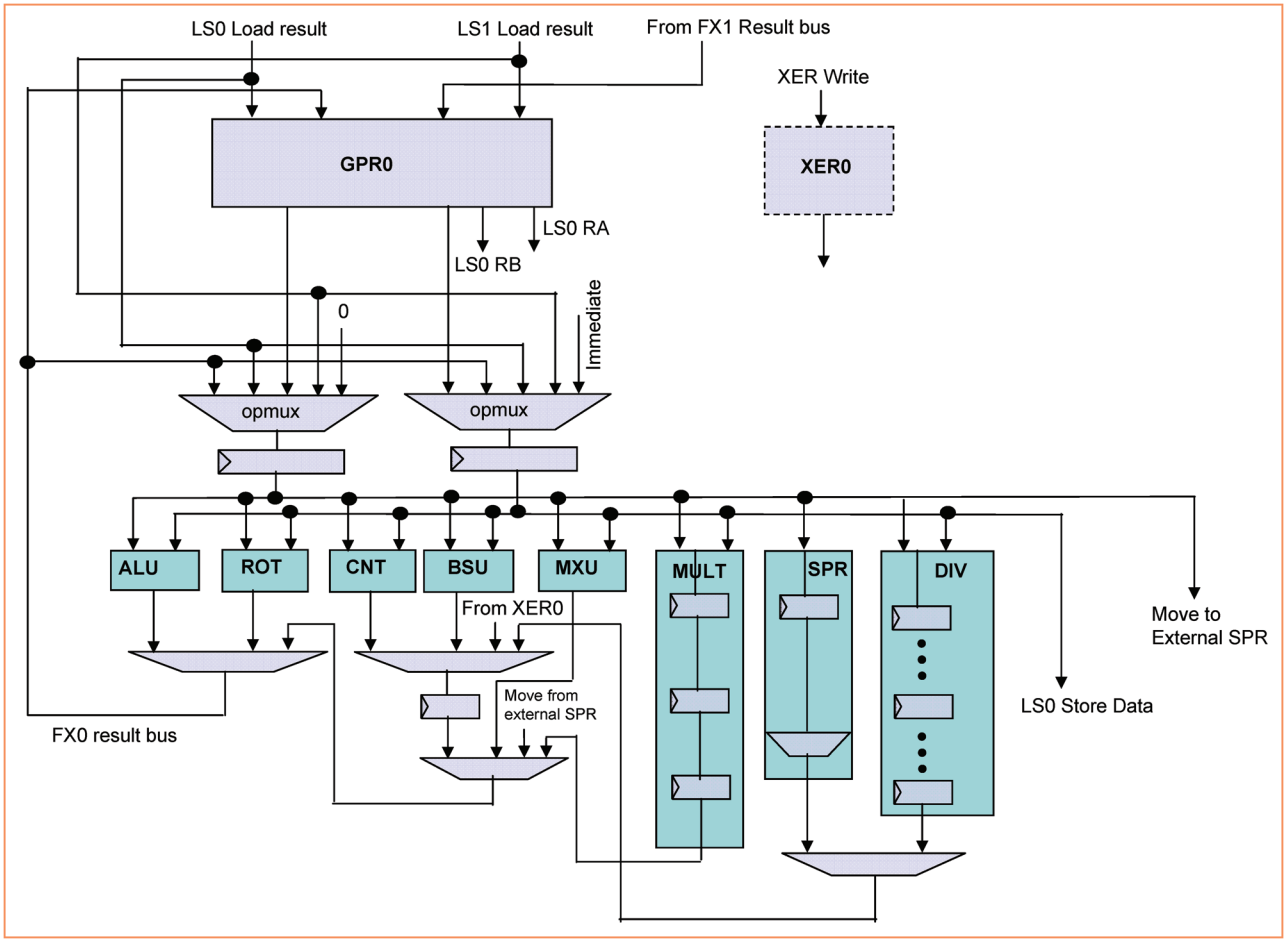


Figure 7

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

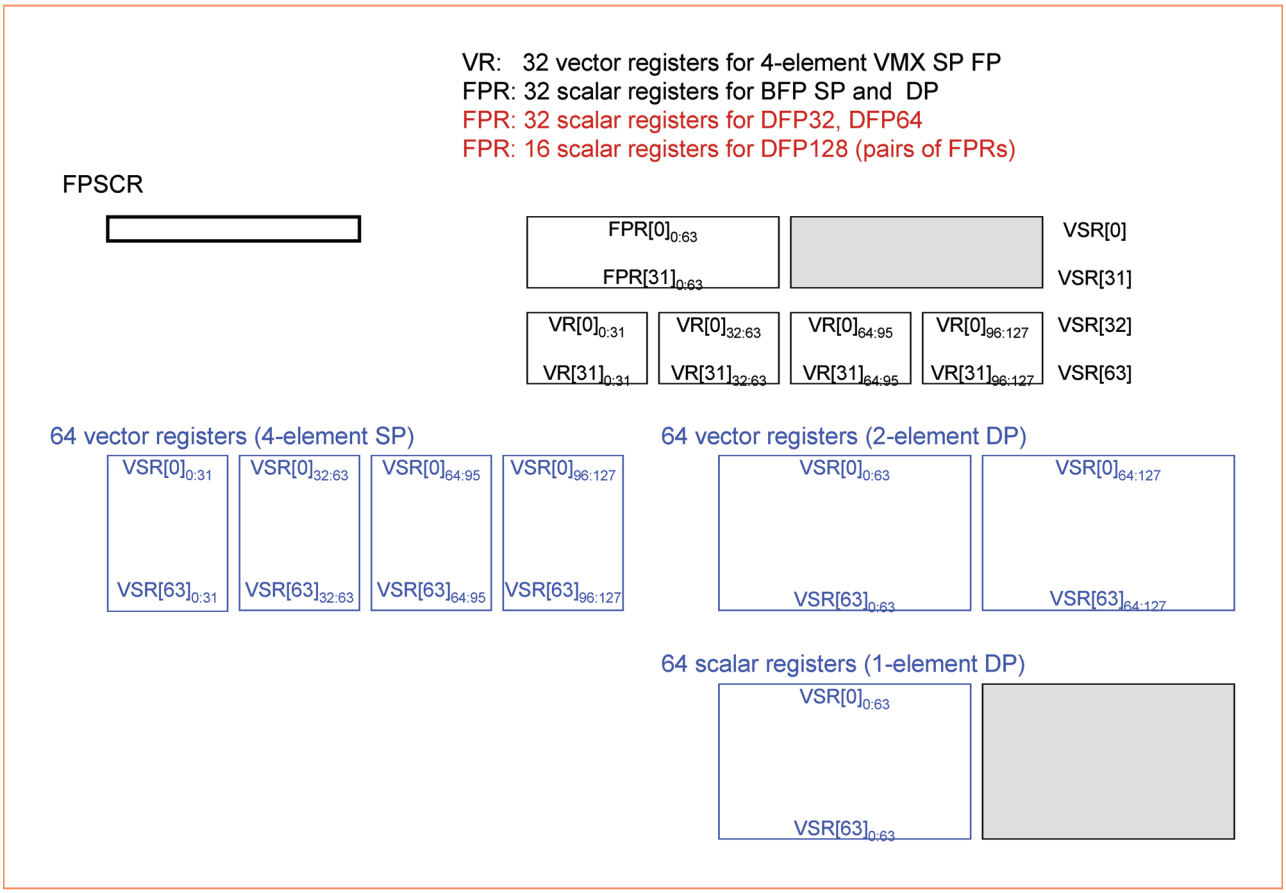


Figure 8

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**.

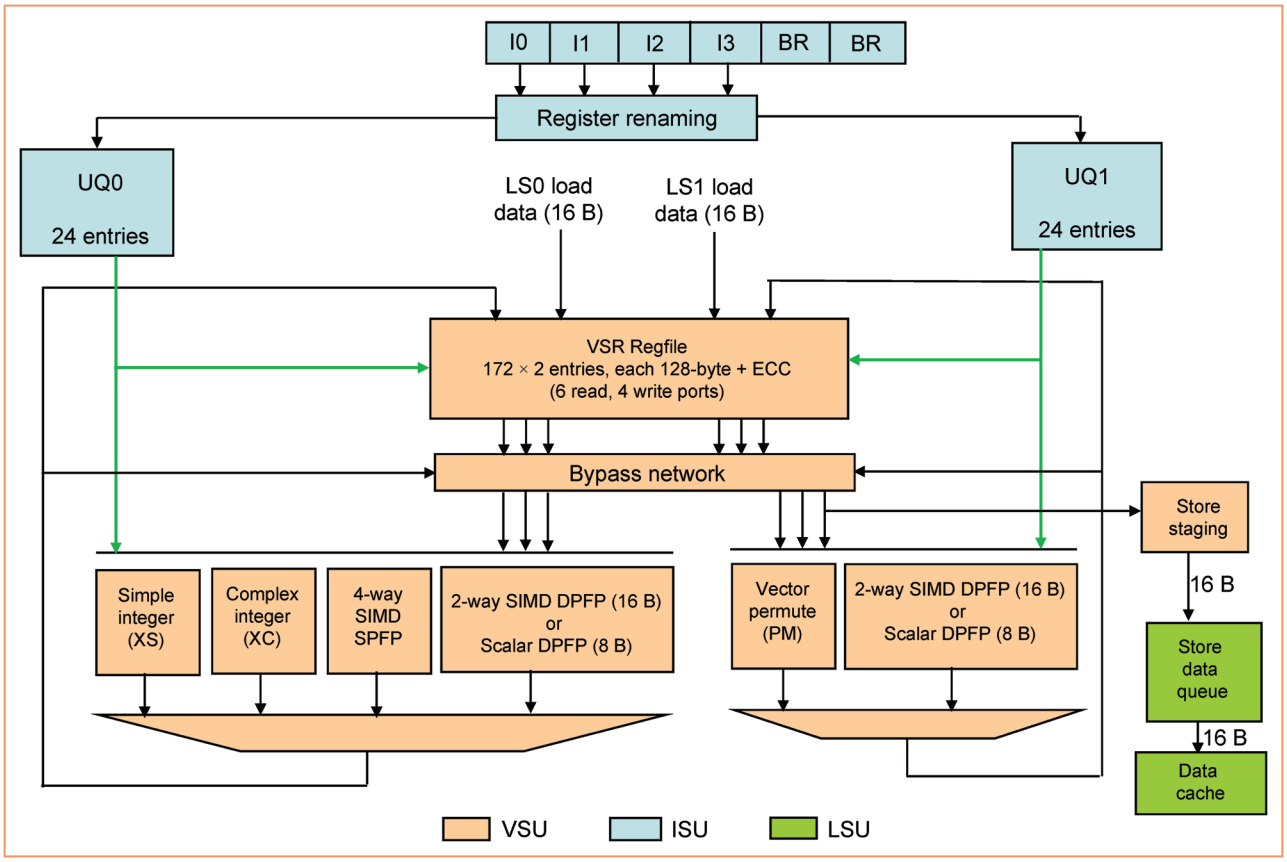


Figure 9

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× 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× a 32-bit or 2× 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.

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

[1]Both 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.

| *State* | *Description* | *Authority* | *Sharers and scope* | *Source data* | *Data cast-out* | *Scope cast-out* |
|---|---|---|---|---|---|---|
| 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.

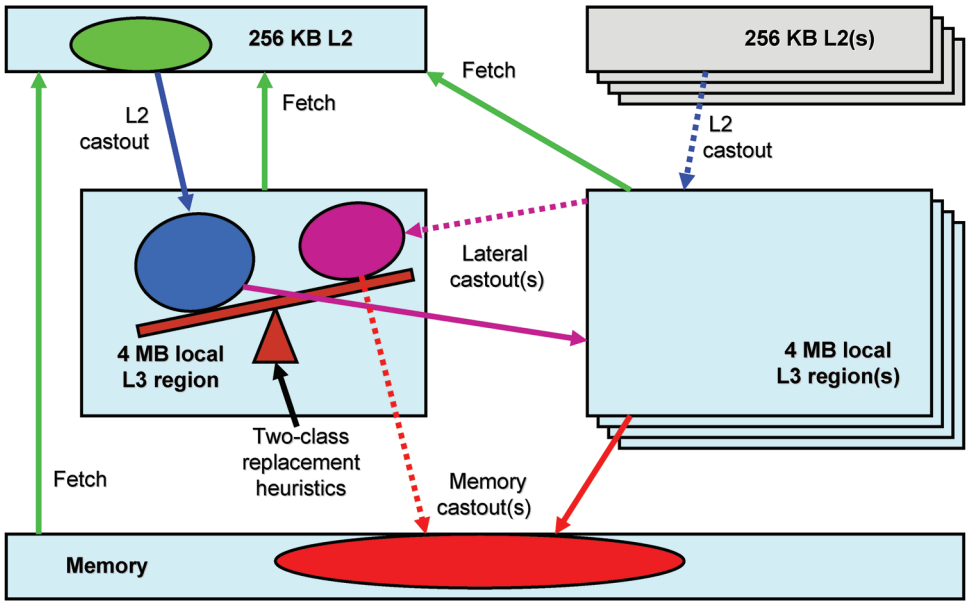


Figure 10

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.

| *DDR3 DRAM frequency* | *Channel frequency* | *Speed ratio* | *Raw channel bandwidth* | *Raw DRAM data bandwidth* |
|---|---|---|---|---|
| 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 multibeat 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 snooper 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 snooper with the lowest possible snoop processing rate. The central coherence arbitration logic must insure that requests (whether sourced from the chip containing the slowest snooper or from another chip in the system) do not overrun the slowest snooper. 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.

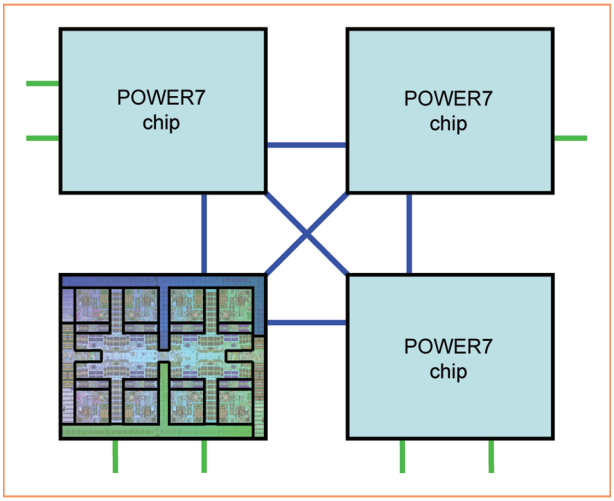


Figure 11

IBM POWER7 first-level nodal topology.

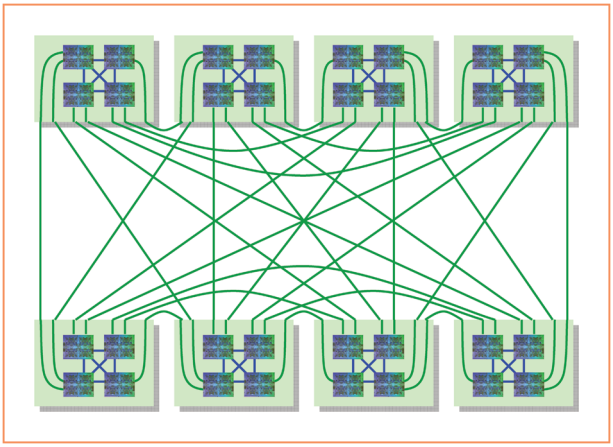

Figure 12

IBM POWER7 second-level system 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 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×, which is less than the 4× to 5× 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 snooper coherence bandwidth. In cases where the speculation is successful, this has the effect of multiplying the link and snooper 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 snooper 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× 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

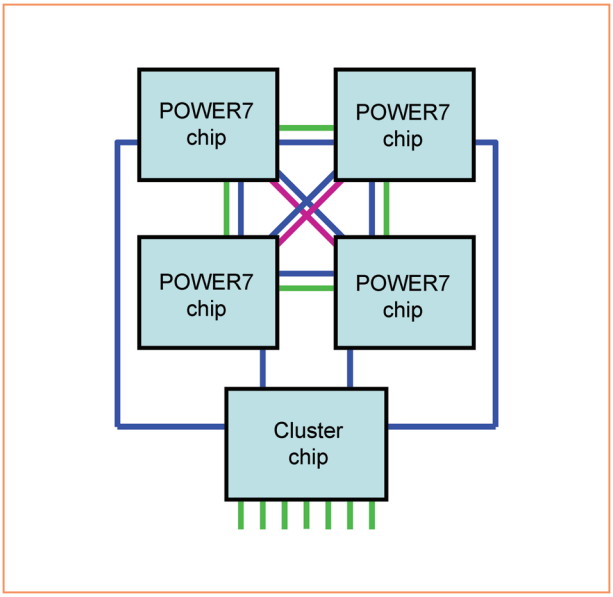


Figure 13

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, availably, 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 |

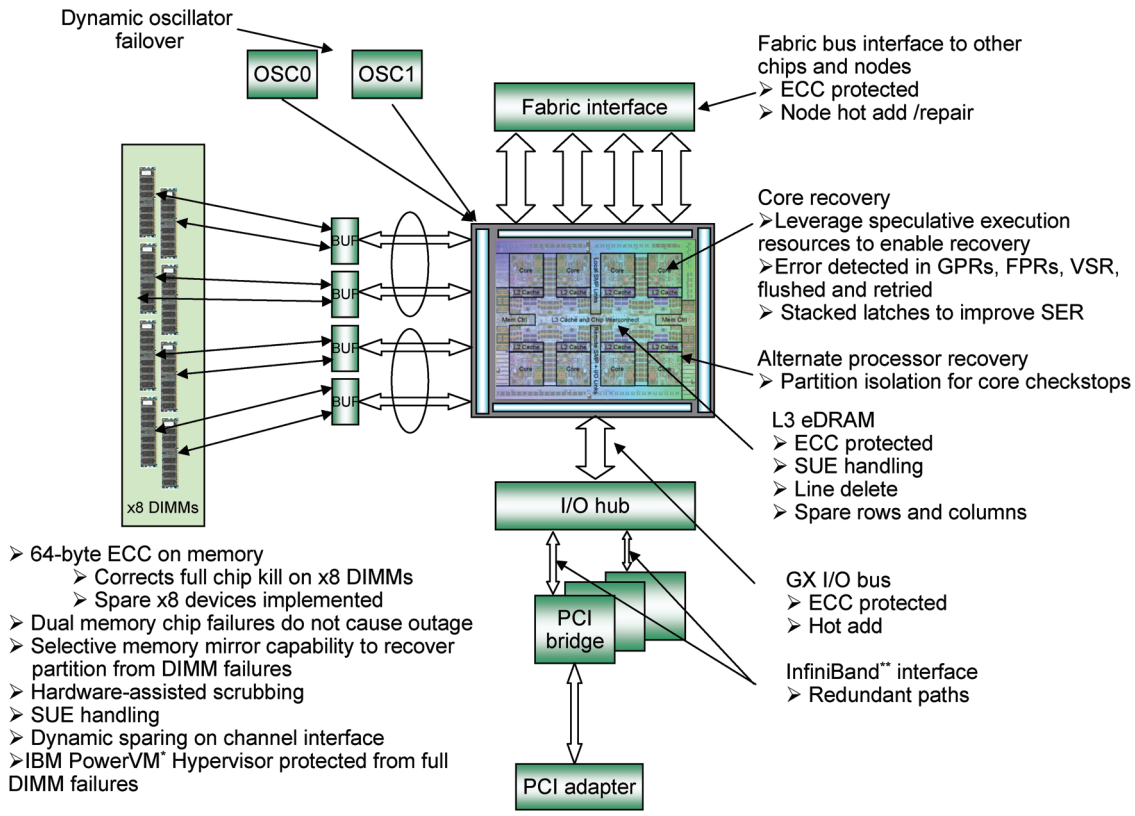


Figure 14

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 checkstop, 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. Hrusecky, 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 (ronchett@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.