Fault-Tolerant Design of the IBM Power6 Microprocessor

The IBM Power6 microprocessor extends the capabilities of the Power5, dramatically increasing its ability to recover from hard and soft errors without increasing system downtime. The Power6 adds new mainframe-like features for enhanced reliability, availability, and serviceability, including instruction retry and processor failover. Optimized for performance and power, the Power6 implements these RAS enhancements without compromising ultrahigh-frequency operation.

Kevin Reick
Pia N. Sanda
Scott Swaney
Jeffrey W. Kellington
Michael Mack
Michael Floyd
Daniel Henderson
IBM Systems & Technology Group

IBM designed and implemented the Power6 microprocessor[1]—from concept phase to system verification—with reliability, serviceability, and availability (RAS)[2] as key goals. RAS is increasingly important in light of current technology trends: increasing silicon device integration and complexity,[3,4] consolidation of applications into fewer processors, and scaled-out computing. As such, RAS characteristics such as availability and data correctness are becoming important metrics contributing to the overall performance of computing systems. It's also important for users to be able to choose a system that meets their requirements for system downtime projection and service cost estimates. Features that relate to the RAS performance at the microprocessor level include the microprocessor's intrinsic reliability, error-detection coverage from a workload perspective, recovery capabilities, cache-error management and recovery, component failover support, failing partition management, and alternate processor recovery.

The Power6, which expanded the Power5's RAS features and implemented additional capabilities from System z, has enhanced core detection and recovery covering nearly all SRAMs and register files. It also includes numerous control and data-flow checkers. At the chip level, enhancements include L2 cache line delete and enhanced error recovery for multibit data errors. Failover lets the processor move workloads to an alternate processor to keep the system available until the next scheduled maintenance. Other new features to Power6 systems include dynamic oscillator failover, dynamic I/O bit-line repair for the dual in-line memory modules (DIMMs), and node hot add and repair.

The Power6 RAS capabilities were made possible by early planning and sharing of System z methodologies, tools, and macros. Error-detection and recovery requirements were defined during the high-level design phase, and the firmware recovery assist requirements were specified early on as well. The RAS design was a close collaboration between the Power6 and System z design teams, enabling the use of concepts initially designed for the System z processor. These features included instruction retry,

Published by the IEEE Computer Society.

alternate processor recovery, and core checkstop isolation.

Power6 microprocessor overview

The new IBM Power6 systems introduce an ultrahigh-frequency processor[1] with a cache hierarchy and memory subsystem that take advantage of the high-frequency multi-threaded dual-core design.

The Power6 processor implements the 64-bit IBM Power Architecture at a 4.7-GHz operating frequency using IBM's 65-nm partially depleted silicon on insulator (SOI) technology with 10 layers of copper interconnect. The 341-mm^2 chip, illustrated in Figure 1, has 790 million transistors. The processor includes two dual-threaded cores. Each core has a private 4-Mbyte L2 cache. The two cores share an on-chip controller for the 32-Mbyte off-chip L3 cache. Also on-chip are dual memory controllers, an I/O controller, and interconnect support for up to 32 chips for a 64-way symmetric multiprocessor (SMP). An integrated SMP switch provides enhanced processor switching capabilities that enable expanded virtualization,[5] expanded RAS, and dynamic processor failover capabilities.

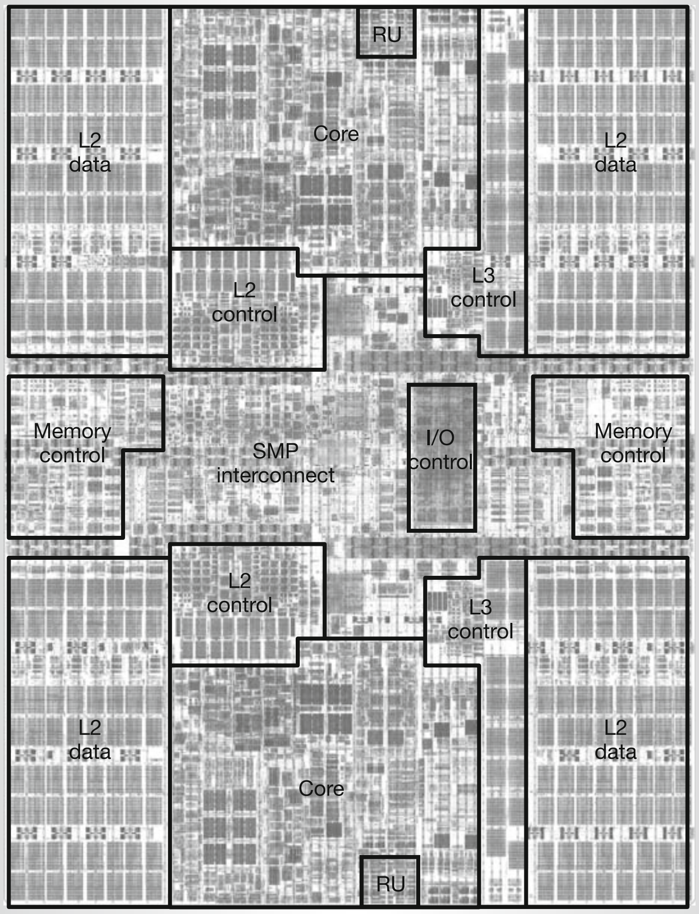

Figure 1. Power6 dual-core microprocessor chip with 8-Mbyte L2, onboard L3 controller, dual memory controllers, and SMP fabric.

Overview of reliability and availability enhancements

As technology continues scaling to higher densities, new silicon error types are emerging and familiar error types are occurring in greater quantities (see the IEEE Workshop on Silicon Errors in Logic: System Effects [SELSE 2007], www.selse.org/program.html). IBM's RAS goal is to minimize the occurrence of errors in customers' systems through robust component design and by offering seamless detection and error recovery. Where microprocessor hardware recovery is impossible, such as in the case of a permanent fault, Power6 can isolate the fault to the failing processor and move the workload to another physical processor (failover).

Figure 2 shows the new reliability and availability features added to the Power6 in boldface type. Features already present in the Power5 (indicated with plain type) include error correction code (ECC) protection of all the signals in and out of the chip to the L3 cache, the memory, the GX bus to the I/O hub, and the fabric bus interface for communication between chips and between nodes. In addition, internal signals are also protected by ECC. L2 and L3 caches and memory are ECC protected. The L1 caches are store-through and protected with parity, such that data stored in the L1 caches are also stored into the L2 cache. When a parity error is detected in an instruction or data cache, the cache is purged and the correct copy refetched from memory. The L3 still has line delete, which purges faulty lines from the logical cache. All caches have special uncorrectable error handling that tags faulty data and flags an

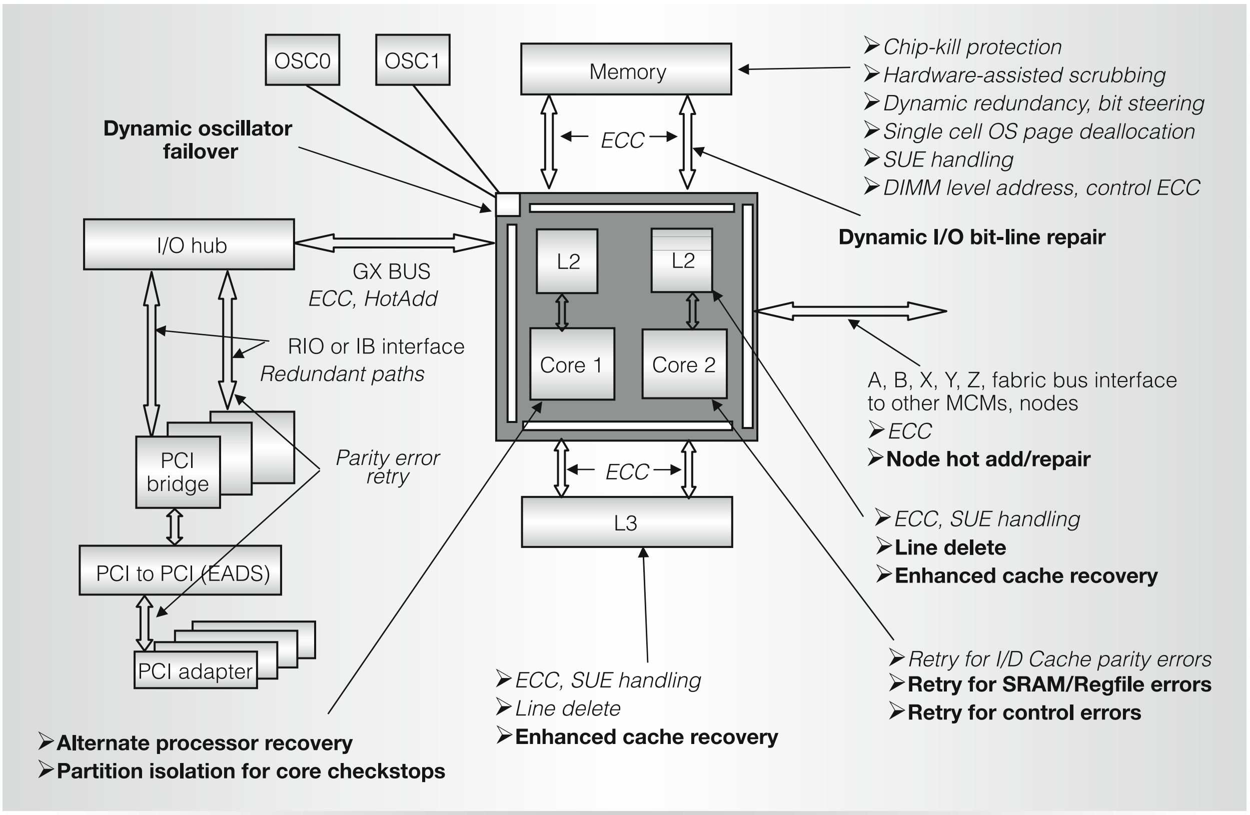
OSC0
OSC1
Memory
Chip-kill protection
Hardware-assisted scrubbing
Dynamic redundancy, bit steering
Single cell OS page deallocation
SUE handling
DIMM level address, control ECC
Dynamic oscillator failover
ECC
Dynamic I/O bit-line repair
I/O hub
GX BUS
ECC, HotAdd
L2
L2
Core 1
Core 2
RIO or IB interface
Redundant paths
A, B, X, Y, Z, fabric bus interface to other MCMs, nodes
ECC
Node hot add/repair
PCI bridge
Parity error retry
ECC
L3
ECC, SUE handling
Line delete
Enhanced cache recovery
PCI to PCI (EADS)
PCI adapter
Retry for I/D Cache parity errors
Retry for SRAM/Regfile errors
Retry for control errors
ECC, SUE handling
Line delete
Enhanced cache recovery
Alternate processor recovery
Partition isolation for core checkstops

Figure 2. New reliability and availability features added to the Power6 processor are in bold. Features already present in the Power5 are in plain type.

error only when the data is consumed. The Power6's memory has all the resiliency features from the Power5, including Chip Kill, hardware-assisted scrubbing, dynamic redundancy and bit steering, single-cell OS page deallocation, special uncorrectable error (SUE) handling, and ECC protection for all interface pins.

Features new to the Power6 include dynamic repair of the memory bus I/Os, dynamic oscillator failover, L2 line delete and enhanced cache recovery, alternate processor recovery and partition isolation for core checkstops, and instruction retry for errors detected in the cores.

Core error handling

The Power4 and Power5 designs limited core error recovery to utilizing the machine check interrupt handler to recover from correctable and uncorrectable errors in SRAMs. Error checkers in the control and data flow produced checkstop conditions and recorded occurrences for fault isolation.[6] The Power6 design includes error-detection logic to most data- and control-flow logic, so it achieves a much higher degree of soft and hard error recovery than previous Power designs. The floating-point units use residue checking, and the majority of latches in the data-flow circuits are protected with parity. The corresponding control circuits are protected by logical consistency checkers, which check that the states are valid with respect to their state machine. Compared to the IBM System z9, which uses replicated execution units operating in lockstep,[7] the inline approach for Power6 error handling consumes less area.

Like the z9, the Power6 uses a recovery unit (RU) that checkpoints the system state every time an instruction completes. The architected state is held in an ECC-protected checkpoint array, which holds the state from both threads. When an error is detected in a core, the RU triggers instruction retry recovery (IRR), and the core restarts from the last checkpointed state. The RU has a pipeline for capturing register results so that the checkpointing lags execution. The checkpoint pipeline must lag execution so that bad results don't enter the checkpoint.

Error-detection facilities throughout the processor core are essential to protect the checkpoint. Fault isolation registers (FIR) report errors from all areas of the processor and logically OR-ed together. Errors that are detected too late to be reported in time to block checkpointing or that affect the checkpoint's restoration are escalated to a processor checkstop. In the latter case, if the checkpoint is intact, restoration on an alternate processor might be possible.

Recovery is performed on a processor core basis, not on an instruction or thread basis. Reported errors don't need to be associated with an instruction boundary as long as they are early enough to block checkpointing. IRR may "back up" to a checkpoint earlier than the instruction that encountered the error. Both threads will be restored from the most recent checkpoint regardless of the error that was detected or the thread on which it occurred.

To make IRR more effective for functional errors, the execution flow through the processor core is simplified (that is, it becomes single-dispatch, nonpipelined) for a short amount of time after recovery. The amount of time in this "slow mode" is controlled by hardware. The recovery is successful when enough instructions execute successfully that the checkpoint advances beyond the simplified execution mode and full-speed execution can resume without error.

The IRR approach is effective for soft errors (such as transient errors induced by radioactive particles or noise) but not for permanent ones. Permanent errors that don't allow the processor to make forward progress are escalated to processor checkstop. Higher-level system recovery enables recovery to an alternate core without taking down the entire system.

Recovery unit checkpoint

Because multiple instructions can complete at the same time in a super-scalar design, all the instructions in the same super-scalar group will checkpoint together. The RU checkpoints the system state after each superscalar group of instructions is completed. on A group tag (GTag) for each thread is sent along at the time of dispatch with the instructions to denote the age of the group relative to each instruction tag (ITag). A group can be checkpointed when the next-to-complete ITag (NTC ITag) is equal to or greater than the GTag.

The fixed-point unit (FXU) and load-store unit (LSU) data must wait at the RU for the floating-point unit (FPU) or vector-multimedia extension unit (VMX) instructions in the same dispatch group to be completed before the whole dispatch group can be checkpointed, because FPU and VMX instructions take more cycles to execute than fixed-point, load-store, or branch instructions. The FPU and VMX instruction results reside in separate queues from the fixed-point instruction results. At dispatch time, the number of FPU instructions expected from the dispatch group, and an indication of whether or not VMX instruction results are expected, are sent along with the dispatching instructions to the checkpointing queues in the RU. The group can only be checkpointed when all the FPU and/or VMX data for that group is available. When a group is checkpointed, the RU indicates to the store queue that any stores from that group can be released to the L2 cache.

Recoverable errors are OR-ed together to block further checkpointing. In order for an error to be recoverable, it must be reported to the RU in time to block the checkpointing of any instructions possibly affected by the error. For some errors, the latency to report them through the fault isolation register (FIR) structure would make them unrecoverable, so they are bundled into "fast-stop" errors and sent directly from each functional unit to the RU. After the recovery state machine completes the reset to clear the error, the RU restores the checkpointed register values to the functional units. The RU has separate buses to the FXU, FPU, and VMX for sending the checkpointed register values, which the functional units wrap back onto the normal functional update buses to restore the value to the working copies.

Recovery sequence

The recovery state machine sequences through a series of operations to recover from an error. First, the machine waits a programmable amount of time to determine if it needs to escalate the error to a checkstop. A secondary error or an error that arrives after the first report might require a checkstop. Then a lookup is done on the GTag in the recovery unit to resolve its mapping to the next instruction address (NIA) that will be executed when the IRR completes. Next, the L2 cache is told to quiesce, and a fence signal is raised to the L2 to indicate that it should ignore all signals coming from the core. All SRAMs in the core then go through array built-in self-test (ABIST) to clear any existing state information. All architected registers are refreshed to the core from the checkpointed state in the RU. Fixed-point, floating-point, and vector registers are routed through separate buses to their respective units. All special-purpose registers (SPRs) are routed through the FXU via the normal functional path used to move SPR instructions. A hardware reset signal is sent to all units to clear any required state that could be corrupted or hung by the error event. The state machine then waits for an ABIST done signal. Next, all FIR bits are reset. All registers are refreshed again. Finally, the fence signal to the L2 is dropped, and the L1 to L2 interface operates normally.

Instruction execution resumes from the restored checkpoint in the reduced-execution mode until a hardware-programmable number of instructions have completed successfully, to ensure that the instruction that caused the original error has been executed. Once the required number of successful instructions have completed, normal operation resumes.

Cache and associated facilities

The L1 cache and its directory support delete mechanisms to work around hard failures. The L1 cache implements the delete on a set boundary, where a cache set is one of *n* locations where a cache line might be placed when it's loaded from the next level in the memory hierarchy. Each cache line in the processor's real memory space can be loaded into any one of the *n* sets. In the Power6, the L1 instruction cache contains four sets and the L1 data cache contains eight.

Because healthy SRAMs will experience soft errors due to normal ambient radiation, the Power6 has a thresholding mechanism to distinguish hard errors from soft errors to invoke a set delete. When an L1 cache or directory parity error is detected, the set identification number is trapped and a bit is set to indicate an error was detected. If a "new" error is detected and the error bit is already active, and the set identification number matches the trapped set identification from the prior error, then it is assumed to be a hard error, and the corresponding set is deleted (marked as unusable). Periodically (every few seconds), the error identification bit is cleared so that a newly detected error will be treated as an independent soft error. Because it might be several cycles from the reporting of a parity error until cache accesses cease, the thresholding takes care not to overindicate (or count more than one error per event) due to multiple parity errors from the same event. When a parity error is detected, indication of a "new" parity error is blocked until the IRR process completes.

The set of deleted lines for associated arrays are tied together (for example, date, directory, and set predict). The LSU or instruction fetch unit (IFU) must indicate when a set is deleted, which set has been deleted, and which array contained the error that caused deletion. Errors from separate arrays set separate FIR bits. The delete indication and set ID is explicitly reported to the core RAS unit and held in a recovery status register. Because a hard error must be detected twice before the set delete is invoked, it's possible to detect the error again without making forward progress after IRR for the first time it was detected. The IRR logic allows for this by implementing a programmable "no forward progress threshold." A counter is incremented in hardware,

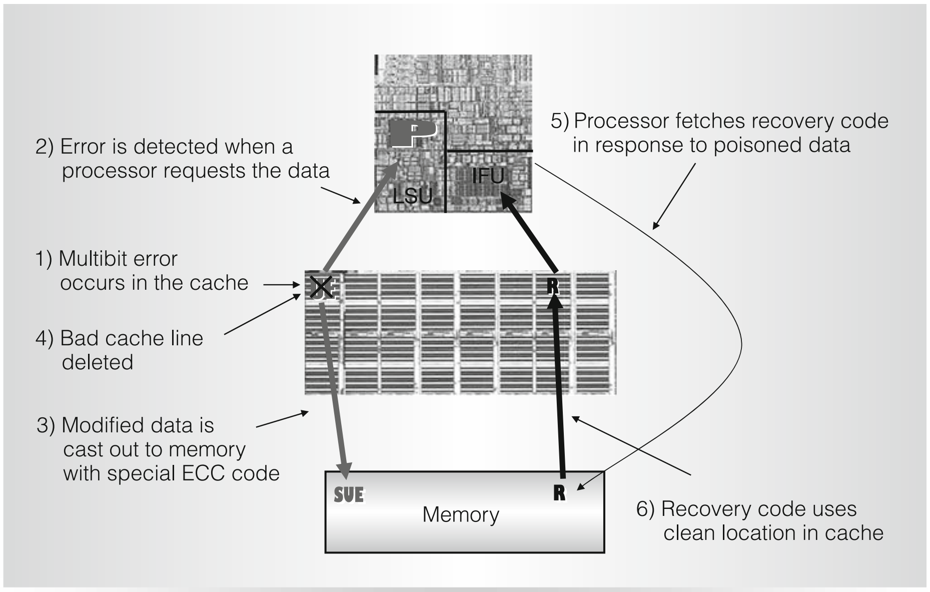
2) Error is detected when a processor requests the data
1) Multibit error occurs in the cache
4) Bad cache line deleted
3) Modified data is cast out to memory with special ECC code
5) Processor fetches recovery code in response to poisoned data
6) Recovery code uses clean location in cache
P
LSU
IFU
X
R
SUE
Memory
R

Figure 3. Uncorrectable-error handling for the L2 and L3 caches.

by a finite-state machine, whenever IRR is executed and forward progress of instruction execution isn't achieved. The finite-state machine will trigger IRR to occur and will increment the counter each time until forward progress is achieved or the counter exceeds the programmable threshold value. When the counter exceeds the threshold value, the processor will checkstop.

Enhanced cache recovery

Uncorrectable errors can originate from memory, cache hierarchy, or interface failures. The PowerPC architecture includes a synchronous uncorrectable error machine check interrupt function. The processor hardware is responsible for trapping the failing storage address for the hypervisor machine check handler. The hypervisor is responsible for containing the damage to the affected workload and utilizing available OS recovery.

Figure 3 shows uncorrectable-error handling for the L2 and L3 caches. The objective is to allow the L2 purge and delete (implemented in hardware) to eliminate uncorrectable errors in unmodified data. The processor goes through normal IRR on the first occurrence of an uncorrectable error, in hopes that it is removed with the purge and delete. While the processor is in the IRR, the L2 purges the line containing the uncorrectable error and deletes the line. Unmodified data is simply invalidated, or modified data will cause a SUE to be cast out to memory.

If the uncorrectable error was in unchanged data, it should go away altogether. If the uncorrectable error was in modified data, it will return (as a SUE) when refetched from memory after IRR and the processor will take a machine check interrupt. The processor fetches recovery code from memory in response to the poisoned data, and recovery proceeds.

Additional enhancements

Power6 has two new redundancy features to boost availability: recovery from bit-line failures and recovery from oscillator failure. Recovery is performed seamlessly to the user.

Dynamic I/O bit-line repair

ECC protects the signals to memory. An additional feature boosting availability is dynamic bit-line repair (see Figure 4). Spare drive and receive wires were added so that if a pin or wire breaks, a correctable error is reported, and data is steered to the spare pin. This happens transparently to the application. The ECC protection lets us

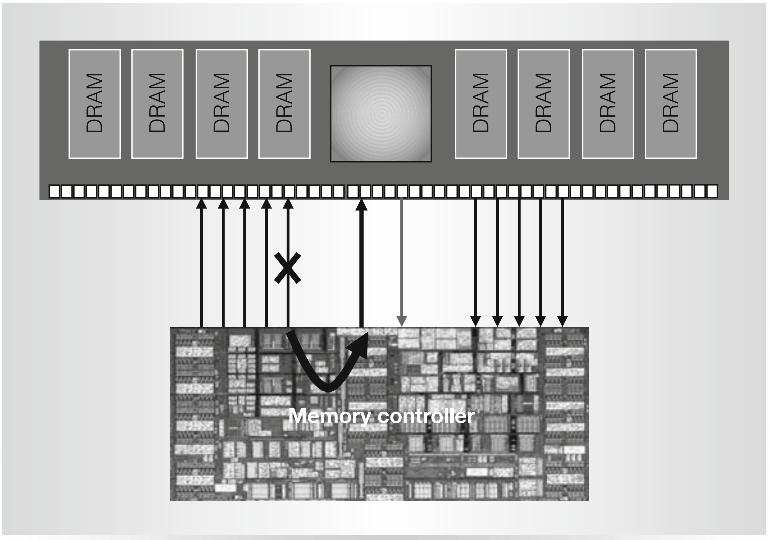

Figure 4. Bit-line repair enables a broken wire to transparently fail over to a spare.

run with a single bit error while the firmware sorts out the failing bit and initiates the switch over. After switch over, there are no further errors. The memory controller to DIMM bit-line failover is sequenced by firmware.

Dynamic oscillator failover

The Power6 contains an oscillator switch macro that receives two oscillator signals. The two oscillators run concurrently. The switch macro tracks the phase of both oscillators and keeps them phase-aligned. If the active clock disappears (fails), the macro will automatically switch to the good clock with no system disruption. The oscillator failover is handled completely in hardware. (see Figure 5).

Beam validation

An effective way to check error resiliency is to use accelerated testing, where faults are randomly injected by irradiation. We performed accelerated studies with protons, neutrons, and alpha particles. The alpha source mimics radiation present in ceramic chip carriers and the solder ball connections. The high-energy proton source mimics the effects from terrestrial neutrons from cosmic rays.

We also performed neutron irradiation studies at the Los Alamos Nuclear Science Center (LANSCE) broad-spectrum neutron source. The proton and neutron results were similar. (Test methodologies and comparisons are available in detail elsewhere.[8])

Here, we can use the case of proton irradiation to validate the Power6's recovery capabilities. While in a prototype system, the microprocessor was irradiated while running architectural verification program (AVP) code that stresses the processor core. All core-related recoveries, checkstops, and AVP errors were recorded while the processor was exposed to the proton beam at the Francis H. Burr Proton Therapy Center.[9]

During those runs, Power6 reported 5,662 events. Of those,

- 5,651 (99.8 percent) were fully recovered and transparent,

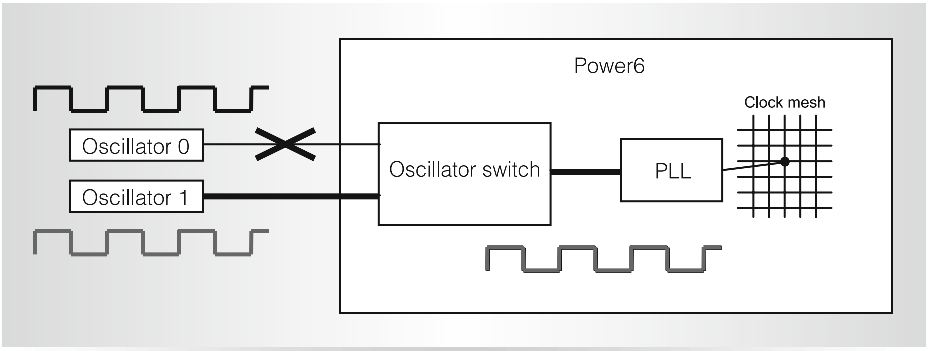

Figure 5. Oscillator failover enables seamless switching from one oscillator to a spare.

- 10 (0.19 percent) resulted in a core checkstop, and
- one (0.01 percent) resulted in a system outage.

We should note that this experiment reports results for one Power6 chip. System data will differ.

The IBM Power6 incorporates mainframe class reliability and availability features without compromising the overall performance, area, and power goals. The result is a 4.7-GHz processor with excellent system performance and scalability that can recover from most errors. Many errors are recovered directly in hardware, and others that result in core checkstop can recover the workload to an alternate processor. A high level of error detection in the core arrays, register files, data flow, and controls, coupled with instruction retry lets the Power6 correct most errors seamlessly. MICRO

Acknowledgments

The Power6 processor was designed, verified, and tested by an exceptional team, and the RAS was enabled by close collaboration with the System z team.

References

1. H.Q. Le et al., ''IBM Power6 Microarchitecture,'' *IBM J. Research and Development*, vol. 51, no. 6, Nov. 2007, pp. 639-662.
2. M.J. Mack, W.M. Sauer and B.G. Mealey, ''IBM Power6 Reliability,'' *IBM J. Research and Development*, vol. 51, no. 6, Nov. 2007, pp. 763-764.
3. S. Borkar, ''Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation,'' *IEEE Micro*, vol. 25, no. 6, Nov.–Dec. 2005, pp. 10-16.
4. R.C. Baumann, ''Radiation-Induced Soft Errors in Advanced Semiconductor Technologies,'' *IEEE Trans. Device and Materials Reliability*, vol. 5, no. 3, Sept. 2005, pp. 305-316.
5. J.M. Tendler et al., ''Power4 System Microarchitecture,'' *IBM J. Research and Development*, vol. 46, no. 1, 2002, pp. 5-26.
6. D.C. Bossen et al., ''Fault-Tolerant Design of the IBM pSeries 690 System using Power4 Processor Technology,'' *IBM J. Research and Development*, vol. 46, no. 1, 2002, pp. 77-86.
7. P.J. Meaney et al., ''IBM z990 Soft Error Detection and Recovery,'' *IEEE Trans. Device and Materials Reliability*, vol. 5, no. 3, Sept. 2005, pp. 419-427.
8. P.N. Sanda et al., ''Power6 Processor Soft Error Resilience,'' *IBM J. Research and Development*, vol. 51, no. 6, Nov. 2008, pp. 662-632.
9. E.W. Casio et al., ''The Proton Irradiation Program at the Northeast Proton Therapy Center,'' *Proc. IEEE Radiation Effects Data Workshop*, IEEE Press, 2003, pp. 141-144.

Kevin Reick is a senior technical staff member at the IBM Systems & Technology Group and is currently the Power7 RAS architect. His work specializes in all aspects of system RAS, system debug design, and virtual bringup. He received his MS in computer engineering from Syracuse University.

Pia N. Sanda is a senior technical staff member with the IBM Systems & Technology Group in the Poughkeepsie-based Technology Development Team and is a member of the IBM Academy of Technology. Her current work focuses on soft error reliability to ensure the dependability of server systems. She received a PhD in physics from Cornell University. She is a senior member of the IEEE.

Scott Swaney is a senior technical staff member with the IBM Systems & Technology Group. He has extensive experience with RAS design at the system and processor microarchitectural levels. He received a BS in electrical engineering from Pennsylvania State University.

Jeffrey W. Kellington is a staff functional verification engineer for the System P Development Group in the IBM Systems & Technology Group. His work includes running Linux and assembly-based functional exercisers and benchmark programs on development hardware and verification

models. He received his BS in computer engineering from the University of Illinois.

Michael J. Mack is a senior engineering manager in processor development for the IBM Systems and Technology Group. His work includes hardware-based processor recovery, SoC design, designer-based verification techniques, and engineering management. He received a BS in computer engineering from Syracuse University.

Michael S. Floyd is a senior engineer with the IBM Server & Technology Group. He specializes in RAS, test, debug, and power-efficient design of server microprocessors and systems. He received a MS in electrical engineering from Stanford University.

Daniel Henderson is a senior technical staff member leading Power Systems availability RAS design. His research interests include Power system hardware design, RAS architecture, test, and fault analysis. He received his BS in electrical engineering from New Mexico State University.

Direct questions and comments about this article to Kevin Reick, IBM Systems & Technology Group, 11400 Burnett Rd., Austin, TX 78758; reick@us.ibm.com.