
POWER4 SYSTEM DESIGN FOR HIGH RELIABILITY

TO ACHIEVE RELIABILITY GOALS, POWER4 SYSTEM DESIGN INCORPORATES FAULT TOLERANCE THROUGHOUT THE HARDWARE, FIRMWARE, AND OPERATING SYSTEM. TOGETHER, THESE SYSTEM COMPONENTS PROVIDE CONCURRENT AND DEFERRED MAINTENANCE, MULTILEVEL RECOVERY FROM ERROR, AND RUNTIME DIAGNOSTICS.

..... Fault-tolerant computing is a mature art whose techniques have migrated from mainframe computers to other product classes. This migration has involved tradeoffs between failure probabilities, defined availability requirements, performance implications, and product cost.¹⁻³ At IBM, we have incorporated fault tolerance in designing Power4 systems—servers comprised of several Power4 chips. The Power4 is an integrated system on a chip (SoC) designed for systems that initially target enterprise Unix customers and at a later date O/S 400 customers. These systems are critical to the successful operation of many organizations and operate 24 hours a day, 7 days a week. Such a high degree of reliability, availability, and serviceability (RAS) is difficult to provide in a highly integrated SoC like the Power4 because traditional error diagnosis techniques become less workable. Therefore, achieving RAS targets for Power4-based systems requires a steady migration of IBM's mainframe S/390 RAS into Power4 system hardware and software.

Description and features

The Power4 chip, currently shipping at 1.1 and 1.3 GHz, contains two independent

processor cores, a shared second-level (L2) cache, a directory for an off-chip third-level (L3) cache, and the required logic to form large symmetric multiprocessors. Figure 1 shows the chip design; it contains 174 million transistors fabricated using IBM's 0.18-micron CMOS silicon-on-insulator technology with seven-layer copper metalization.

Each Power4 processor is an out-of-order superscalar design with eight execution units:

- two fixed point,
- two floating-point,
- two load/store,
- a branch unit, and
- an execution unit that performs logical operations on the condition register.

Each processor can issue instructions to each execution unit every cycle, although the maximum instruction retirement rate is five per cycle. Each core also contains a 64-Kbyte L1 instruction cache and a 32-Kbyte L1 data cache with dual ports. Up to eight concurrent data cache misses and three instruction cache misses are possible. The superscalar pipeline can simultaneously support more than 200 instructions in various stages of execution.

Douglas C. Bossen
Joel M. Tendler
Kevin Reick
IBM

The two cores share an eight-way set-associative L2 cache organized around three independent cache controllers. At 1.3 GHz, 125 Gbytes/s of data can go from the L2 cache to the two processor cores. The Power4 chip can handle a total of 12 concurrent outstanding L2 misses.

The on-chip L3 directory supports an off-chip eight-way set-associative 32-Mbyte cache, which supports up to eight outstanding L3 misses. Data transfer between the L3 cache and the Power4 chip occurs at 13.9 Gbytes/s at an operating frequency of 1.3 GHz. Transfers to and from memory can occur at up to 12.8 Gbytes/s.

Four Power4 chips are mounted on a single module to form an eight-way system, as Figure 2 shows. Figure 3 (next page) shows the interconnection of four such modules to form a 32-way system. From a chip perspective, the interconnect topology is bus based; from a module perspective, it is switch based. The interconnection between modules is ring-

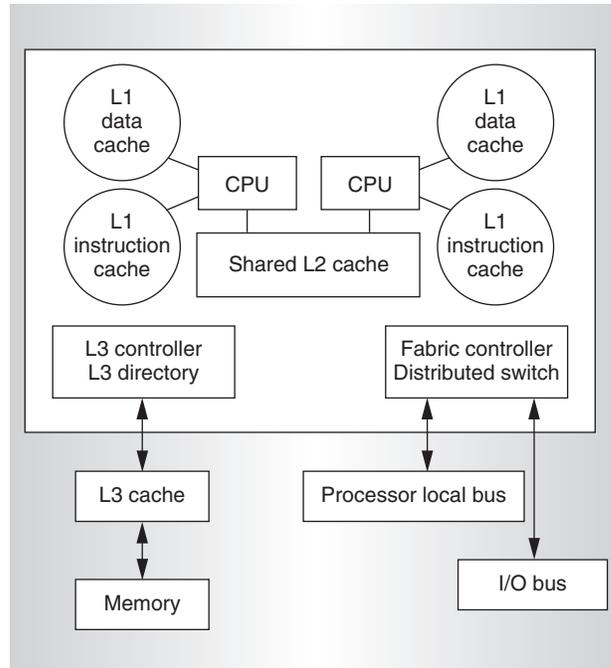


Figure 1. Power4 chip design.

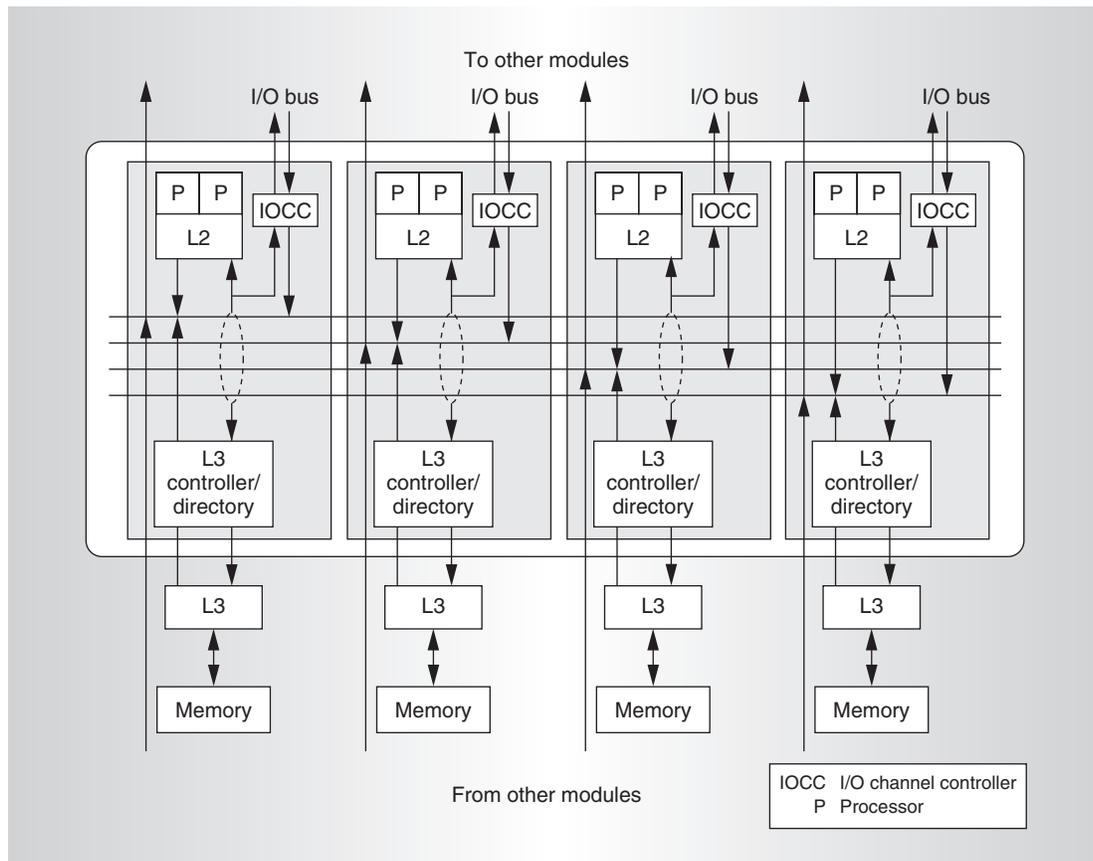


Figure 2. Power4 multichip module with four chips.

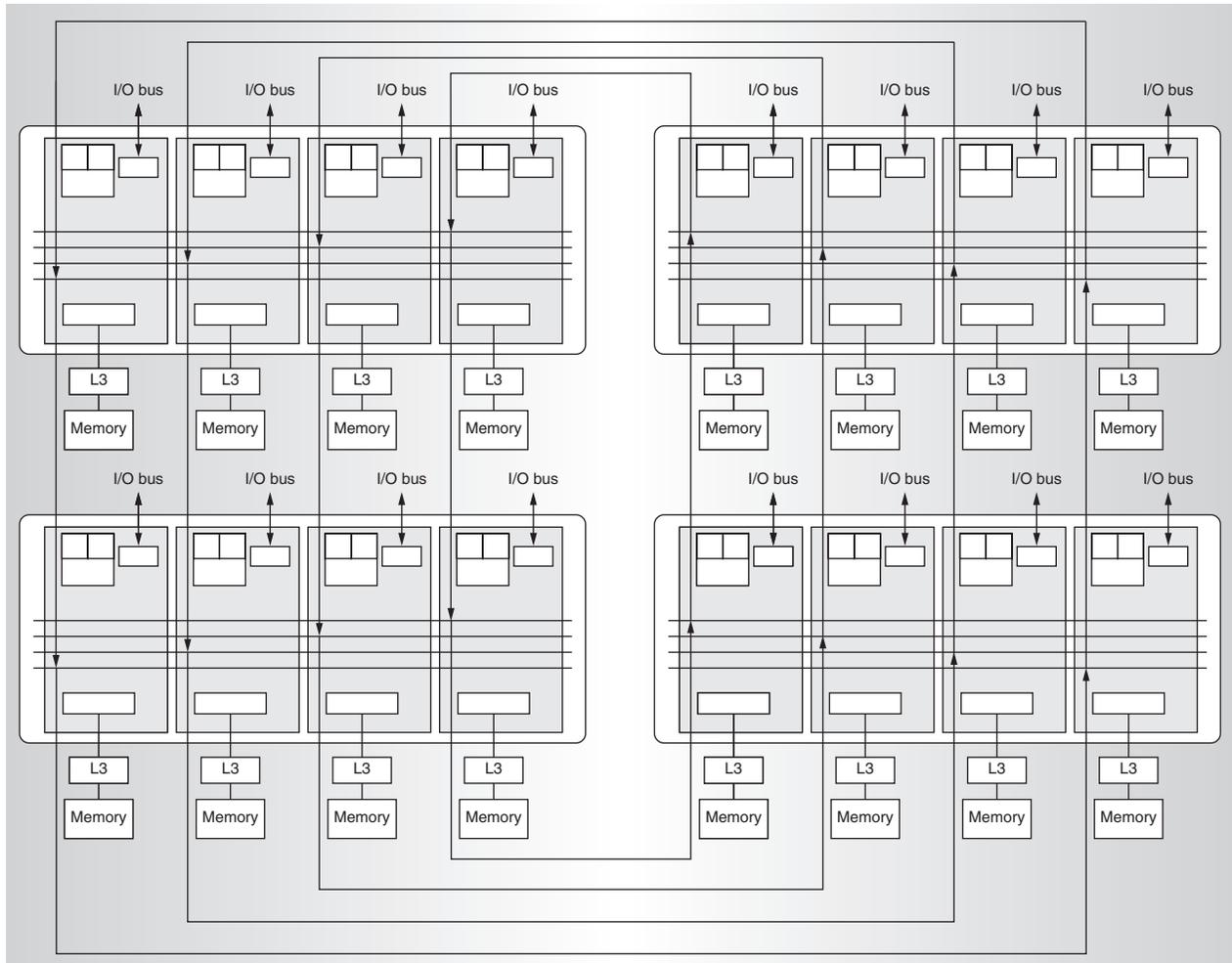


Figure 3. 32-way Power4 system.

like. Power4-to-Power4 buses, whether on or off module, operate at 650 MHz—half the processor speed. Buses to and from an off-chip L3 cache operate at 433 MHz—one-third the processor speed. You can further interconnect multiple Power4 nodes in a cluster configuration to form even larger systems.

System reliability requirements

Power4 reliability targets include classical mean time between failure (MTBF) improvements and customer impact availability improvements, called high-impact outages (HIOs), related to recoverable and unrecoverable failures. Enhancing and adding new retry mechanisms, increasing the amount of redundant hardware, and increasing the degree of concurrent and deferred maintenance compared to prior systems have improved reliabil-

ity. In fact, Power4 systems reduce the number of HIOs per year by more than a factor of 2 over predecessor systems such as the IBM eServer p680 and the IBM/SP (based on the Power3-II microprocessor).

To achieve MTBF and HIO improvements, Power4 systems have functions that support

- fault masking,
- internal redundancy,
- error recovery, and
- dynamic reconfiguration for all system elements, including the processors, system memory and caches, I/O adapters and devices, storage devices, and power and cooling elements.

Table 1 lists the reliability features employed

Table 1. Power4 system reliability features.

Feature	Depends on
Refresh error correction for L1 cache	Hardware
Error correction code protection for L2 and L3 caches, to protect against single-bit errors	Hardware
Chip-kill coverage so that the system can fully recover data stored in a failed memory chip	Hardware
Correctable error thresholds for main memory and L1, L2, and L3 caches, so that the system can automatically schedule a service call before a hard error occurs	Hardware and firmware
Redundancy to replace faulty hardware with extra hardware built into the system	Hardware and firmware
Dynamic deconfiguration for L1 and L2 caches	Hardware and firmware
CPU dynamic deconfiguration	Hardware, firmware, and operating system
Dynamic cache line delete on L3 cache	Hardware
Internal system bus error recovery for multichip module fabric, I/O bus, and peripheral component interconnect (PCI) bus parity error	Hardware, firmware, and operating system
Partition or process containment for system unrecoverable errors	Hardware, firmware, and operating system
Redundant bit steering at initial program load (IPL) for L1 and L2 caches	Hardware and firmware
Redundant bit steering and bypass mode at IPL for L3 cache	Hardware and firmware
Persistent deconfiguration for CPUs, caches, and memory	Hardware and firmware
Concurrent fail-over and repair for redundant I/O drawer link	Hardware and firmware
Simple-access PCI hot swap	Hardware and firmware
Runtime first-occurrence diagnostics (basic and extended)	Hardware and firmware

in Power4 systems to achieve these HIO goals. All depend on hardware functions, and most require system firmware and operating system support for the complete solution.

Cache recovery for soft errors

In Power4 CMOS technology, the ability to tolerate and mask soft errors in all caches is paramount. If cache errors were not recovered, aggregate soft-error rates (SERs) in the SRAMs would cause unacceptable customer outages. These SERs exceed the conventional intrinsic failure rate in such technology by up to a factor of 5,000. To alleviate the effect of soft errors, Power4 provides error correction at all cache levels. In the L2 and L3 caches, Power4 uses Hamming error correction code (ECC) for standard single-error correction and double-error detection.

The L1 data cache is parity protected, through *inclusivity* (all data in the L1 data

cache is guaranteed to also be in the L2 cache) and a *store-through design* (all writes to the L1 data cache are also written to the L2 cache). System firmware can accomplish error recovery by flushing the affected cache line and refetching the data from the L2 cache. The Power4 has a similar recovery function for data arrays—the translation look-aside buffer and the effective-to-real address table (ERAT)—to assist and speed up the translation of addresses from effective (what the application programmer deals with) to real addresses (where the data is actually stored in memory).

For the L1 instruction cache and its ERAT, the hardware reports parity errors as if a cache miss had caused a refetch from the L2 cache. In the L2 ECC implementation, correct data is always written back into—and is then refetched from—the L2 cache. This differs from most main-memory ECC implementations. In this case, the corrected data goes to

the fetch requester; it is not rewritten into main memory until a special write operation, such as scrubbing, occurs.

Recoverable error thresholds and internal redundancy

The most likely failure event in a processor is a soft single-bit error in one of its caches. However, other events can occur, and it's important to distinguish these events from one another. For the L1, L2, and L3 caches, hardware and firmware track whether the particular ECC station corrects permanent errors beyond a certain threshold. After exceeding this threshold, the system creates a deferred-repair error log entry. The system also initiates additional runtime availability actions, such as *CPU vary off* (which initiates a controlled shut-down of a processor such that no information is lost) and *cache line delete* (which removes a faulty cache line—consecutive addresses in the cache—from service).

Both L1 and L2 caches in the Power4 chip come with spare bits in their arrays. Programmable steering logic permits access to these arrays to replace faulty bits. This programmable logic is analogous to the redundant bit steering employed in main memory as a mechanism to avoid physical repair. For the L1 and L2 caches, the steering logic is activated by built-in self-test (BIST) at power on.

The Power4 design implements L3 cache redundancy at the granularity level of a cache line. Exceeding correctable error thresholds during runtime invokes a dynamic cache-line-delete function capable of up to two deletions per L3 cache. In the rare event of solid bit errors exceeding this quantity, the cache continues to run, but the service processor issues a message calling for deferred repair. Rebooting the Power4 system without such repair places the L3 cache in bypass mode; the system then comes up with this cache *deconfigured*. Performance suffers without the L3 cache, but the system still operates—a more acceptable alternative than an inoperable system.

Dynamic CPU deconfiguration

Firmware records runtime error thresholds, indicating the number of corrected, permanent errors in both L1 and L2 caches. These errors become part of the system error log as deferred-maintenance service events. If the sys-

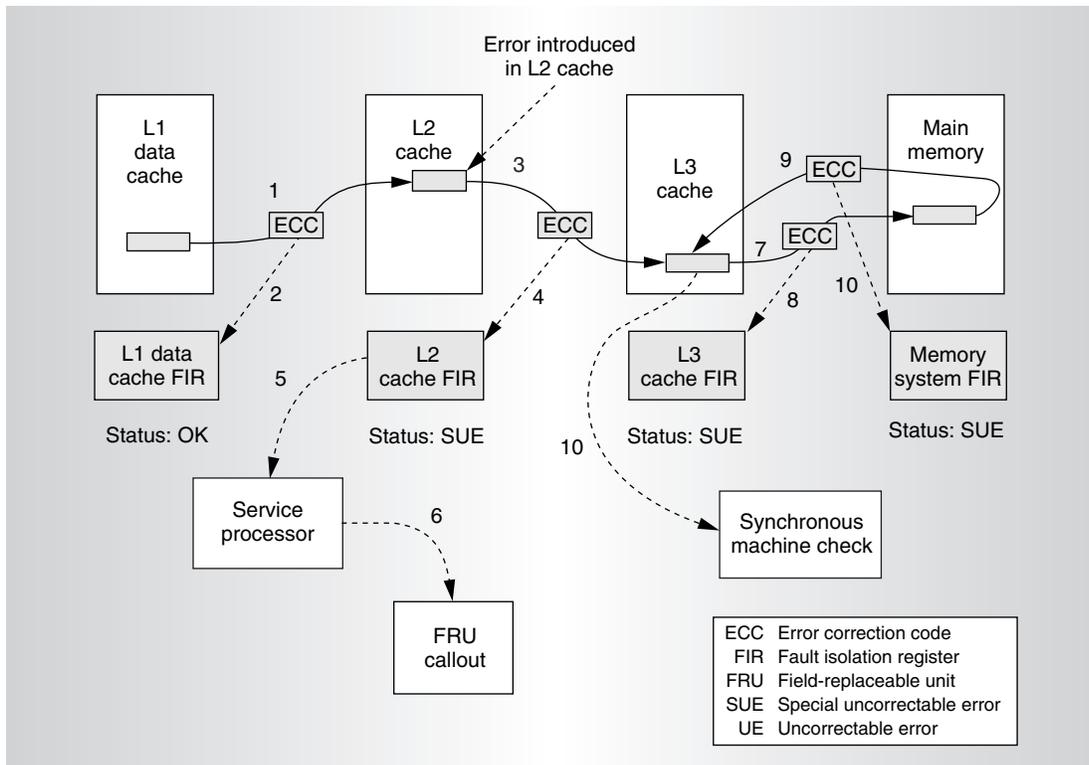
tem administrator chooses, these events can invoke a runtime firmware procedure called **stop_self**. This procedure purges the caches of the affected processors and marks them as deconfigured to the operating system. Deconfigured processors do not respond to or cause any system events; removing these processors from the configuration provides continued system operation until a deferred repair.

Multilevel recovery

The Power4 supports noncheckstop behavior in the unlikely event of corrupt data from various sources. It does so by sending a synchronous machine-check interrupt to system software in response to load data having either a parity error from the L1 cache or an unrecoverable error from the L2 cache. The firmware's machine-check interrupt handler examines the Power4 error state and determines the appropriate response. For L1 parity, the firmware initiates recovery. For an uncorrectable error, the machine-check handler returns the address of the instruction that loaded the corrupt data. Operating system recovery involves determining the nature of the affected process and initiating user-process or partition termination, depending on whether the error damaged the kernel. Only the affected partition is terminated in a multiple-partition system.

Uncorrectable-error handling

Power4 error-handling recovery has multiple layers to maximize system availability. ECC throughout the system and *bus retry* (retrying an operation on a bus for which an error was detected) accomplish recovery for the bulk of anticipated errors. In the rare event that errors exceed the ECC capability, or bus retry fails to provide correct data, predecessor machines would respond with some form of system checkstop—that is, halting machine operation—to maintain data integrity. For Power4 systems, we designed the hardware, firmware, and AIX (Advanced Interactive Executive) to keep running—avoiding a hardware checkstop but maintaining data integrity. The Power4 tags corrupt data—that is, data with uncorrectable errors (UEs)—with various hardware indicators: a special uncorrectable error (SUE) syndrome in memory, a Derr signal for buses, special error packets for



I/O transmissions, and so on.

Figure 4 shows the sequence of events surrounding an uncorrectable error. The processor concurrently writes data to the L1 and L2 caches (step 1). An ECC station checks the data on the bus and reports it to the L1 data cache's fault isolation register (FIR) as OK (step 2). In this example, a UE occurs while the data is still in the L2 cache. When the L2 cache casts out the data to the L3 cache (step 3), an ECC station checks the data and reports a UE to the L2 FIR (step 4). The L2 FIR in turn reports the error to the service processor (step 5), which calls for a replacement of the faulty part (step 6). An ECC station writes the data into the L3 cache, which encodes it as an SUE to indicate that its error condition has already been reported to the service processor. When the L3 cache casts out the data to main memory (step 7), an ECC station notes the SUE state and records it in the L3 FIR (step 8). The SUE state inhibits the report of this change to the service processor. System error handling occurs only when corrupt data is referenced (step 9). As soon as the processor recognizes corrupt data, it signals a synchronous

machine-check interrupt (step 10), indicating to the operating system the address of the instruction loading the corrupt data. In a multiple-partition system, only the partition referencing the corrupt data will have an outage. If the data error does not affect the kernel, only the user process is terminated. As with many errors, the focus is on minimizing the error's impact and keeping as much of the application operational as possible.

ECC design for handling uncorrectable errors

As data moves through the memory hierarchy, it is repeatedly checked and corrected at each level to ensure both data integrity and fault isolation. To support UE handling, all ECC stations extend conventional ECC. Any level of the hierarchy can store known corrupt data that originated outside its level, giving it a special tag for fault isolation so that on subsequent requests the data will be recognized as corrupt. The mechanism used in all Power4 ECC stations lets the ECC logic modify the check bits using a fixed algorithm before storing known corrupt data. Then, in subsequent fetches, a unique ECC pattern is recognized

as a stored SUE, distinct from unrecoverable double-bit errors. If a normal double-bit unrecoverable error is encountered, the ECC logic forwards the corrupt data to the next memory level, marked as an SUE with appropriate check bits. Only the initial observation of incorrect data causes fault isolation capture. The check-bit modification algorithm guarantees that an SUE stored in a location having a legitimate single-bit error will not alias as a correctable error; the SUE will still be recognized as an unrecoverable error on subsequent fetches. This SUE state inhibits reporting of previously reported error conditions.

Runtime diagnostics based on error checkers

All Power4 error-checking mechanisms—including parity detection, ECC, and control checks—have three distinct but related attributes:

- Checkers provide data integrity.
- Checkers initiate appropriate recovery mechanisms—from bus retry based on parity error detection, to ECC correction based on hardware detection of a non-zero syndrome in the ECC logic, to firmware refreshing of certain arrays based on parity detection.
- All ECC stations in Power4 data and control paths deterministically isolate physical faults based on runtime detection of each unique failure that could occur.^{4,5}

Figure 5 shows the error checker placement in a typical 32-way Power4 system. All error checkers contain software-readable error-capture FIRs and blocking logic so that for every detected error, only the first checker that encounters the error records it. This form of instantaneous runtime diagnostics greatly enhances other forms of diagnostic testing, such as BIST (which relies on reproducible defects rather than intermittent ones), that are often present or evident only at runtime. Runtime error diagnostics are deterministic, defining and documenting the checker's error domain at every check station. This function also allows automatic deconfiguration of a failed element without testing or recreating the failure. Diagnostic validation includes dynamically injecting intermittent error con-

ditions during runtime to verify that the diagnostic identifies the physical component where the error resides.

Debugging solutions

To enhance system bring-up in the laboratory and to continue making forward progress in the presence of what otherwise could be catastrophic failures, Power4 systems implement *hang detection* in both the processor and the storage subsystems. During system bring-up (before shipping the system to customers and during test), the hardware could have a fatal error that would cause the system to cease functioning—a *hang*. Hang detection lets the system detect that it has ceased functioning so that corrective action can be taken to circumvent the failure and avoid a catastrophic failure. Continued testing would still be possible, reducing overall test time. We adjusted the hang detection intervals so that the processor has a longer time-out than the storage unit processing the request. The idea is to force a hang error closer to the fault's actual source. This strategy allows a more accurate diagnostic call and simplifies debugging of the hang. You can think of the hang interval as an inverted triangle with the processor at the top and the I/O at the bottom.

We took advantage of Power4's processor microarchitecture to implement a hang recovery mechanism. Historically, complex out-of-order processors have encountered hang situations due to internal resources mismanagement. The hang detector in each Power4 processor, however, can distinguish whether the source of the hang is due to internal processor operations or a lost storage request. Backing up the processor and retrying the instruction stream in a slow mode, where the instructions are completed in order, permits recovery from internal processor hangs. After a predetermined time interval, the processor returns to full speed. We used this mode of hang recovery during Power4 bring-up to work around design errors.

IBM's experience with large complex systems has shown the need, under certain circumstances, for internal machine data well beyond the surgically limited FIR data needed for normal diagnostic field-replaceable unit (FRU) fault isolation. In these rare but critical situations, IBM engineers need trace arrays, debugging buses, and scan-ring dump

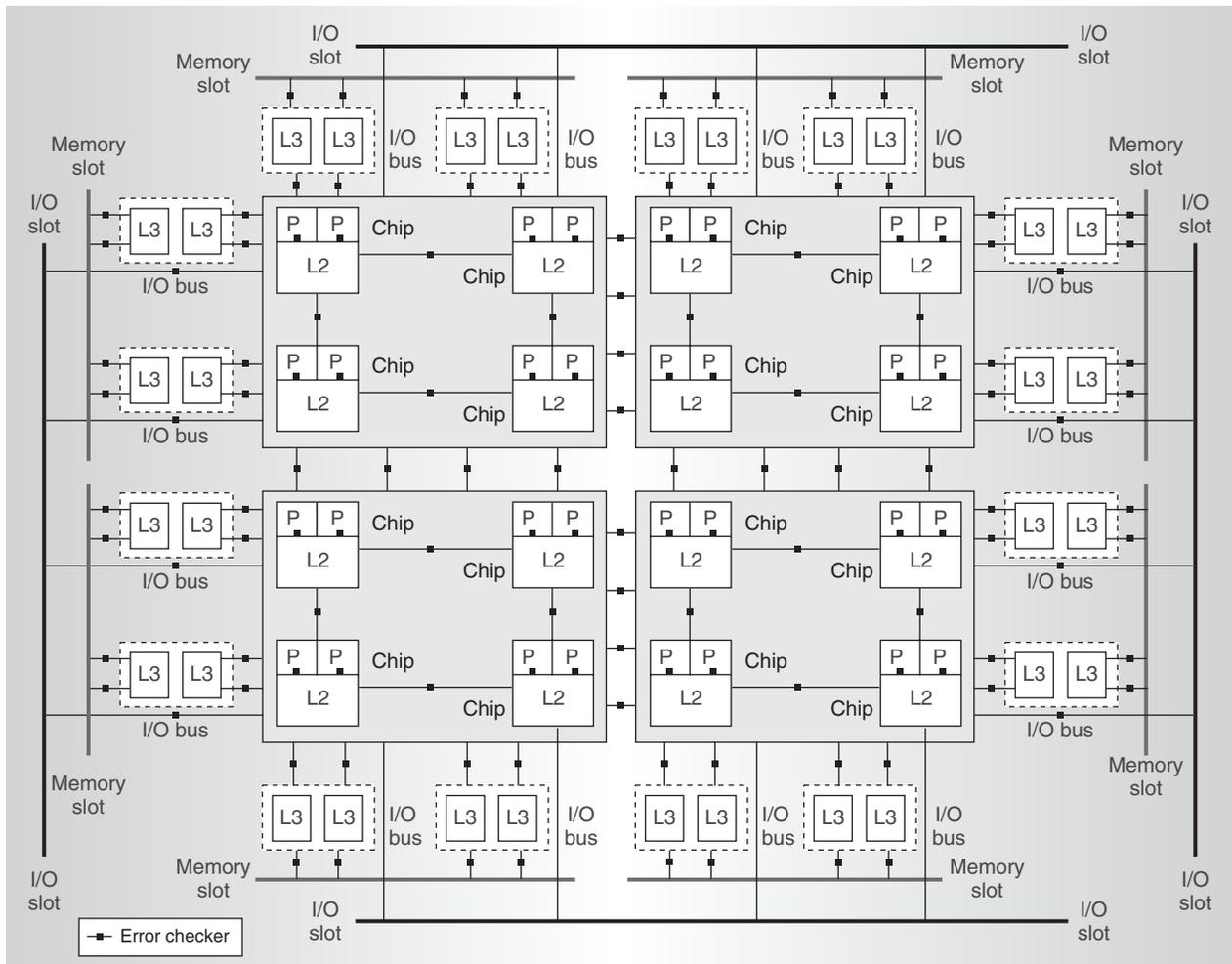


Figure 5. Error checkers for fault isolation.

access to determine the root cause as rapidly as possible. Power4 hardware has hardware hooks to acquire this data, if necessary. At the system level, only certain error checkers—those indicating time-outs and hangs not addressed by recovery and not having a clear FRU callout—automatically cause the long scan-ring dump to occur, complete with trace contents defined by the hardware design team.

As is always the case with recovery, there is more to do. Over time, we will be enhancing Power4-based systems with additional RAS functions. In fact, IBM has an initiative in place, called Project eLiza, to further enhance system reliability and automate the management of information systems we develop. The items included in Power4 described in this article are a start in this direction.

MICRO

References

1. C.L. Chen and M.Y. Hsiao, "Error-Correcting Codes for Semiconductor Memory Applications: A State-of-the Art Review," *IBM J. Research and Development*, vol. 28, no. 3, Mar. 1984, pp. 124-134.
2. D.C. Bossen, L.C. Chang, and C.L. Chen, "Measurement and Generation of Error Correcting Codes for Package Failures," *IEEE Trans. Computers*, vol. 27, no. 3, Mar. 1978, pp. 203-207.
3. C.L. Chen et al., "Fault-Tolerance Design of the IBM Enterprise System/9000 Type 9021 Processors," *IBM J. Research and Development*, vol. 36, no. 7, July 1992, pp. 765-779.
4. N.N. Tendolkar and R.L. Swann, "Automated Diagnostic Methodology for the IBM 3081 Processor Complex," *IBM J. Research*

and Development, vol. 26, no. 1, Jan. 1982, pp. 78-88.

5. D.C. Bossen and M.Y. Hsiao, "Model for Transient and Permanent Error Detection and Fault-Isolation Coverage," *IBM J. Research and Development*, vol. 26, no. 1, Jan. 1982, pp. 67-77.

Douglas C. Bossen is a distinguished engineer at the IBM Server Group, Austin, Texas. His research interests include computer hardware fault-tolerant design, RAS, and migration of cost-effective RAS techniques into the pSeries and iSeries server brands during product development. Bossen has a BS, MS, and PhD in electrical engineering from Northwestern University. He is a fellow of the IEEE and a member of Sigma Xi.

Joel M. Tendler is program director of technology assessment at the IBM Server Group, Austin, Texas. His research interests include assessing emerging technologies for applicability in future eServer, iSeries, and pSeries product offerings. He was also the lead performance analyst on the Power4 project. Tendler has a BS in engineering from the Cooper Union, and a PhD in electrical engineering from Syracuse University.

Kevin Reick is a senior technical staff member at the IBM Server Group, Austin, Texas. His research interests include systems; storage; service processor and microprocessor architectures; architecture for RAS design; system debugging; and design for testability. Reick has a BS in electrical engineering from Rutgers University, and an MS in computer engineering from Syracuse University.

Direct questions and comments about this article to Douglas C. Bossen, IBM Server Group, 11400 Burnet Road, Austin, TX 78758; dbossen@us.ibm.com.

For further information on this or any other computing topic, visit our Digital Library at <http://computer.org/publications/dlib>.

Coming Next Issue

MAY-JUNE 2002

Unorthodox Computer Architectures

Guest Editor Karl E. Grosspietsch (Fraunhofer Institute of Autonomous Intelligent Systems, Germany) present articles about Unorthodox Computer Architectures. Topics will include

- An Intelligent Memory Architecture for the Support of Artificial Neural Nets,
- A Parallel Neural Processor for Real-Time Processing Applications,
- SDAARC: An Extended COMA Architecture, and
- ULSI Architectures for Artificial Neural Networks.

IEEE Micro
serves your
interests

IEEE
micro