Fault-tolerant design of the IBM pSeries 690 system using POWER4 processor technology

by D. C. Bossen
A. Kitamorn
K. F. Reick
M. S. Floyd

The POWER4-based p690 systems offer the highest performance of the IBM eServer pSeries™ line of computers. Within the general-purpose UNIX® server market, they also offer the highest levels of concurrent error detection, fault isolation, recovery, and availability. High availability is achieved by minimizing component failure rates through improvements in the base technology, and through design techniques that permit hard- and soft-failure detection, recovery, and isolation, repair deferral, and component replacement concurrent with system operation. In this paper, we discuss the fault-tolerant design techniques that were used for array, logic, storage, and I/O subsystems for the p690. We also present the diagnostic strategy, fault-isolation, and recovery techniques. New features such as POWER4 synchronous machine-check interrupt, PCI bus error recovery, array dynamic redundancy, and minimum-element dynamic reconfiguration are described. The design process used to verify error detection, fault isolation, and recovery is also described.

1. Introduction

The IBM eServer pSeries* p690 is a general-purpose server consisting of a number of units, including a processing unit with CPUs, caches, memory, bus control elements, and a service processor element, a power and cooling distribution unit, and I/O drawer units. The processing unit utilizes POWER4 technology.

The POWER4 chip, targeted for frequencies higher than 1 GHz, contains two independent processor cores, a shared L2 cache, an L3 directory, and all of the logic needed to form large SMPs. The chip, containing more than 170 million transistors, is fabricated using IBM 0.18-μm CMOS SOI technology with seven-layer copper metallization. Each POWER4 core 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 to perform logical operations on the condition register. Instructions can be issued to each execution unit every cycle, although the maximum instruction retirement rate is five per cycle. Each core also contains a 64KB L1 instruction cache and a dual-ported store-through 32KB L1 data cache. Up to eight data- and three instruction-cache misses are supported. More than 200 instructions can exist concurrently in various stages of execution. The two cores share an eight-way set-

0018-8646/02/$5.00 © 2002 IBM

associative unified L2 cache organized as three independent cache controllers. More than 100 GB/s of data can be moved from the L2 to the two processor cores. In aggregate, 12 outstanding L2 misses can be supported by the L2. The on-chip L3 directory supports an off-chip eight-way set-associative 32MB cache that supports up to eight outstanding L3 misses. Data transfer between the L3 (and the memory behind it) and the POWER4 chip is in excess of 10 GB/s. POWER4 chips can be mounted on either single-chip modules or multichip modules to form larger SMP systems.

Up to four POWER4 chips can be mounted on a single module to form an eight-way system. Four such modules can be interconnected to form a 32-way system. From a chip perspective, the interconnect topology is bus-based. When viewed from a module perspective, it is switch-based. The interconnection between modules is ringlike. POWER4-to-POWER4 buses on and off module operate at half the processor speed. Buses to and from an off-chip L3 and memory operate at one-third the processor speed. Multiple POWER4 nodes can be further interconnected in either a cluster or a NUMA (nonuniform memory access) configuration to form even larger systems.

The subject of this paper is the fault-tolerant design used in the p690 system to detect, recover, and isolate failures throughout the system. Fault-tolerant design affects two figures of merit vital to customers: availability and duration of repair. A key design goal of the p690 system is to provide high levels of reliability, availability, and serviceability (RAS).

At the broadest level, the p690 system is designed to provide the highest levels of availability and data integrity in the presence of hardware failures. To provide these levels of fault tolerance, the hardware, the AIX* operating system, and the internal service and recovery firmware have been designed with special characteristics to support fault tolerance. All system hardware components, including logic, storage arrays, power, and cooling, have significantly increased levels of detection, isolation, error correction, recovery effectiveness, and degraded or bypass operation.

Factors such as hardware-intrinsic reliability as well as recovery do help in reducing machine repairs and outages. Hardware-intrinsic reliability for the p690 has been provided by careful component selection in order to reduce as much as possible the opportunity for failure, achieving a significant reduction over predecessor systems. The remaining leverage is derived from painstaking component-level design for fault masking and recovery.

Built-in hardware error detection and FRU isolation [1, 2] have been an integral feature of IBM mainframes beginning with the IBM 308X and continuing through the IBM 3090*, ES/9000*, and zSeries* systems. In the pSeries, this capability was introduced in 1997 in the F50, S70, and all following servers. It has been successfully used in all of these platforms to quickly isolate failures to a failing field-replaceable unit (FRU). A key design characteristic which zSeries and pSeries share, and which is also used in the p690 design, is to a) detect errors during normal machine operation, b) capture machine status information at the time of error detection, and c) isolate the failing FRU by analysis of the data captured at the time of detection of the error. Fault-isolation design is discussed in Section 2.

2. Run-time self-diagnostic fault isolation

The diagnostics goal for the p690 system is to isolate 95% of the failures to a single FRU. For 5% of the failures, two FRUs plus any boards or wires that interconnect the FRUs are candidates for fault identification. In these 5% of cases, manual isolation procedures may be employed by the service person.

The role of error checkers in diagnostics

All POWER4 error-checking mechanisms, including parity, ECC, and control checks, have three distinct but related attributes. First, checkers provide data integrity. Second, checkers initiate appropriate recovery mechanisms, from bus retry based on parity error detection, to ECC correction based on hardware detection of a nonzero syndrome in the ECC logic, to firmware-executing recovery routines based on parity detection. Third, and equally important, all error-check stations have been placed in POWER4 system data and control paths to deterministically isolate physical faults based on run-time detection of each unique failure that may occur. All error checkers are instrumented with software-readable error-capture fault-isolation registers (FIRs) and blocking logic, so that for every detected error only the first checker that encounters the error records it. This form of instantaneous run-time diagnostics greatly enhances other forms of diagnostic testing, such as built-in self-test (BIST), which relies on reproducible defects rather than the intermittent ones often present or evident only at run time. Run-time error diagnostics are deterministic in that for every check station, the unique error domain for that checker is defined and documented. Diagnostic validation consists of dynamic run-time injection of intermittent error conditions, to determine that the correct physical component is called out by the diagnostic.

Let us consider FRU isolation. It is desirable to confine 100% of failures to a single FRU to minimize service cost and customer impact. This is not always possible [3] in a multi-FRU system, for reasons given below. In a multi-FRU system, the board provides wires for inter-FRU communication of data-bus, address, and control signals. When a bad signal is received on the receiving FRU, it is not possible to tell whether the driver on the sending

FRU has failed, the receiver on the receiving FRU has failed, or the board wire that interconnects them has failed. Hence, when an error is detected on an inter-FRU communication path, the failure is classified as isolated to two FRUs. The percentage of failures for which two FRUs are called can be kept to a minimum by checking the signal on the sending FRU and checking the received signal on the receiving FRU. Roughly 5% of the total number of circuits on a FRU are used for inter-FRU communication, so the design permits calling two FRUs plus the board for 5% of failures and calling a single FRU for all remaining failures.

Next, we describe two design techniques that are used to achieve a high percentage of single-FRU isolation. We also describe the role played by the service processor in FRU isolation. Finally, the fault-isolation process is described.

Design techniques for FRU isolation

To isolate failures to a single FRU, it is necessary to minimize situations in which failure of a circuit on one FRU is detected by a checker on another FRU. Two techniques, or design rules, are used to accomplish this: a) rules for checking of signals that cross FRU boundaries, and b) use of the active source identifier as part of the detected and captured error state. These techniques are described below.

Checking of signals crossing FRU boundaries

An example of using error checkers at FRU boundaries to improve and define the optimum FRU isolation is shown in **Figure 1**. Signals to be sent originate at a parity-checked register on FRU 1. They enter combinational logic, array, or gating logic. Error checker E is used to isolate the failure of the combinational logic circuits to a single FRU. When these circuits fail, checkers D and E are off and checker F is on.

Active source identifier

In addition to the identity of the error checker that detected an error, certain machine-state information must be captured along with the error checker name to facilitate FRU isolation. The specific information, collectively called the active source identifier (ASI), was first introduced in the IBM 308X [1] and 9021 processors and is briefly explained here. It is used to point to the source of the data in a situation for which one source out of many possible sources supplies the data that is checked.

An example of an active source identifier is shown in **Figure 2**. Register A or Register B could be the source of the data that is sent to Register C. A latch (ASI) points to the source of the data. Thus, the ASI contains, for example, a 0 if data came from Register B and a 1 if data came from Register A. Consider data transfer from Register B to Register C. The ASI would be 0. If parity checker C detects an error, the failure is isolated to

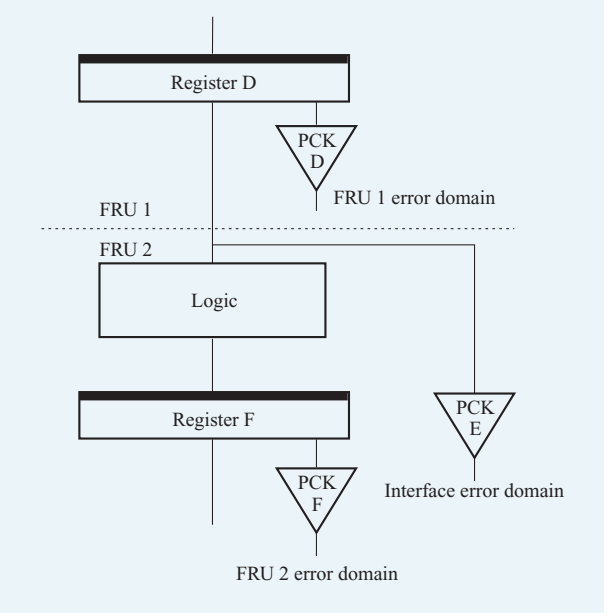

Figure 1

Error checking at FRU boundaries (PCK = parity checker).

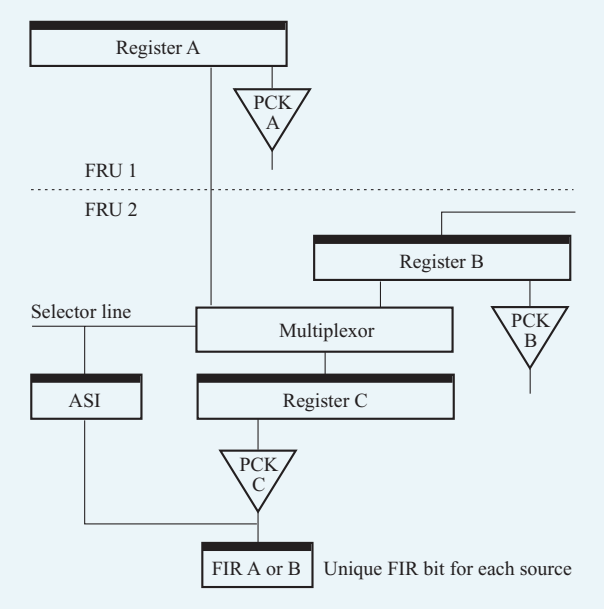

Figure 2

Example of an active source identifier (ASI).

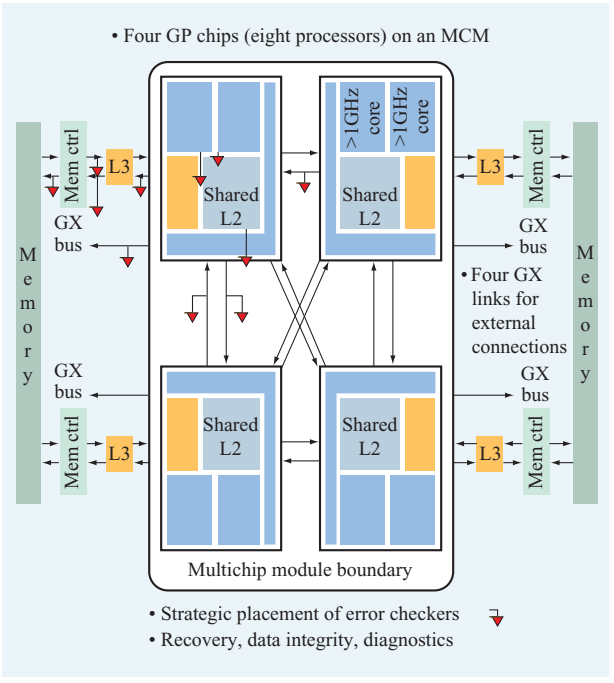

Figure 3

Logical structure of POWER4 multichip module.

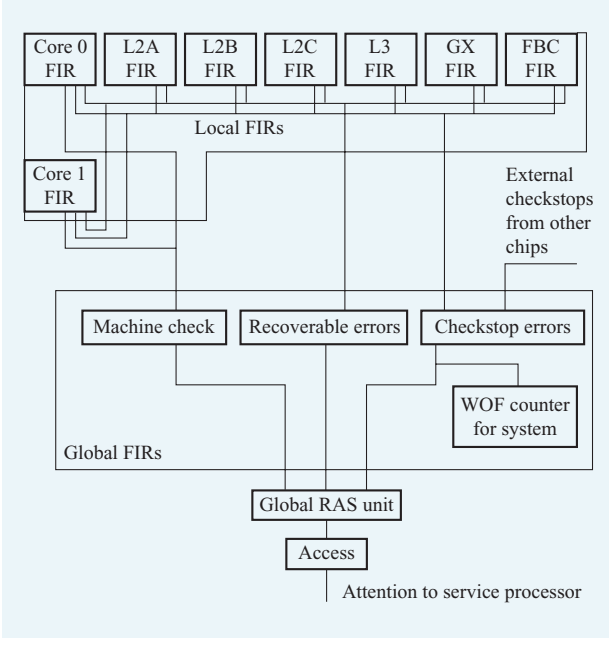

Figure 4

Summary diagram of FIR internal check stations.

FRU 2. In the absence of the ASI, both FRU 1 and FRU 2 would have to be called when parity checker C comes on. In the p690 implementation of the ASI concept, the ASI information would be combined directly with the error checker C output, so that one of two different fault-isolation register (FIR) bits would be set, depending on the data source. This allows simpler analysis of the error state by the processor runtime diagnostic (PRD) firmware, since extra state bits do not have to be examined by the firmware in most cases to obtain the minimum diagnostic resolution. These internal hardware functions are defined by the diagnostic team item by item, often three years before system deployment, early in the chip development of any new chip set.

Who's on first logic structure

For every error checker there is a latch which is set to ON when the checker detects an error. Prior to design closure on every hardware element, each error checker is examined as to the domain of failures which can cause the checker to come on. This analysis takes into account the total p690 system set of error checkers, including special hardware which determines which error checker came on first.

The logic structure which supports this procedure is sometimes called "who's on first" (WOF), and is depicted in **Figure 3**. The WOF limits the domain of any error checker backward in the data or control flow to the previously checked signal source, and resolves any ambiguity due to error propagation. It is implemented in every POWER4 chip. To keep the wiring within reasonable bounds, the WOF structure is actually implemented as a hierarchy of global, MCM, and chip-internal FIR registers, which are examined sequentially to determine the source domain of an error. In particular cases, the WOF counters on each chip are examined in order to pinpoint which chip first detected an error, and which FIR points to the source error. Ultimately, the domain becomes the FRU call. If design analysis reveals that a chip function has ORed two different error checks, one with a single FRU domain and the other with a domain spanning two or more FRUs, the design is changed so that each such checker has a separate latch rather than sharing a latch with one or more other checkers. This critical hardware analysis is performed on all p690 and pSeries hardware, so that the self-diagnostic FRU call is deterministic, not requiring manual interpretation.

System error-checker placement

The diagram in **Figure 4** shows a high-level summary of the internal check stations that feed the various levels of the FIR structure to support the p690 full-spectrum diagnostic design.

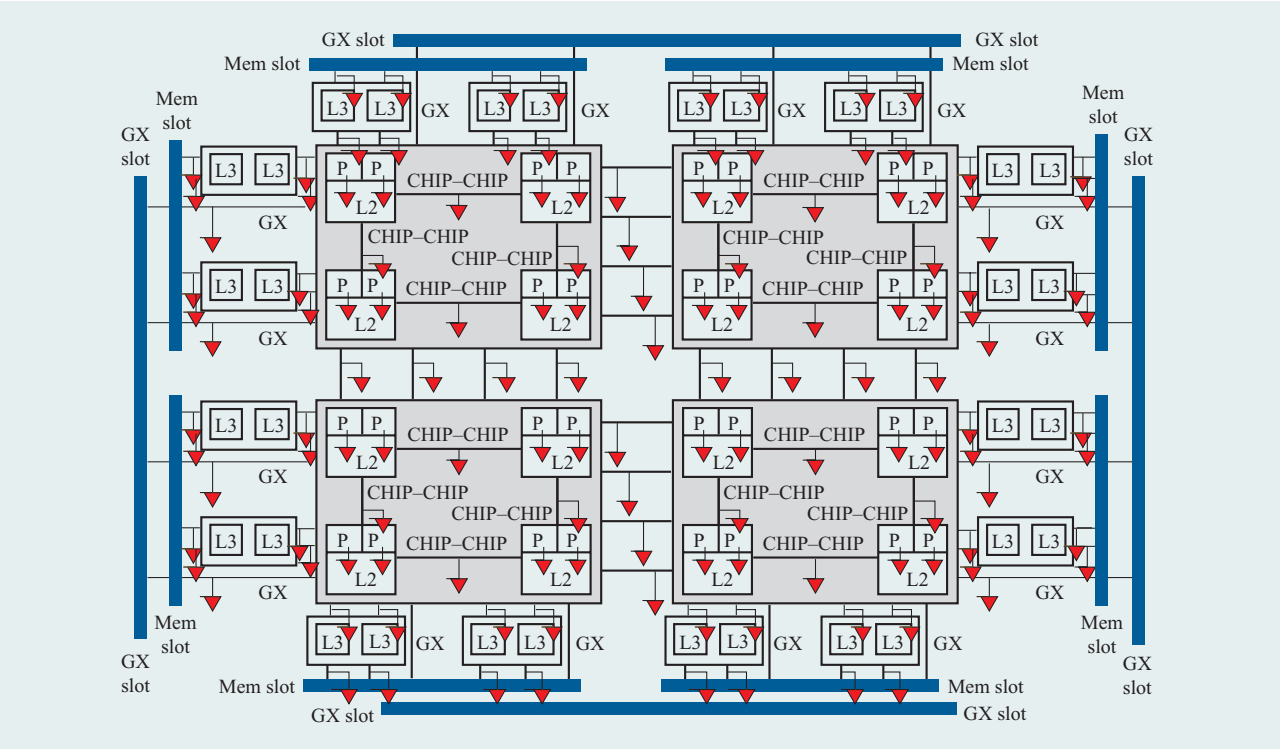

Figure 5

Placement of error checkers in the p690 system.

Design for chip isolation internal to the FRU

The level of diagnostic resolution of most error checkers in IBM products such as the p690 is generally more precise or refined than a FRU, even though the FRU resolution is sufficient for field-repair purposes. For example, the CPU FRU for the p690 is a multichip module (MCM) containing eight processor cores and four L2 caches, on four separate chips. Each of these chips has separate and unique FIR registers, so that the actual diagnostic resolution of a detected error is a single chip. The chip call information at time of failure is permanently written into a separate hardware module returned to a manufacturing facility along with the replaced MCM. This allows root-cause analysis of the defect based on the exact error state, including chip and exact checker, without resorting to recreating the failure, and is a key element of IBM's continuous hardware field-quality monitoring. **Figure 5** indicates error-checker placement in a typical 32-way p690 system, for the purpose of detailed diagnostic resolution. (Error checkers are denoted as in Figure 3.)

Role of the service processor in FRU isolation

The role of the service processor in FRU isolation is similar to that in IBM 308X, 3090, and 9021 machines.

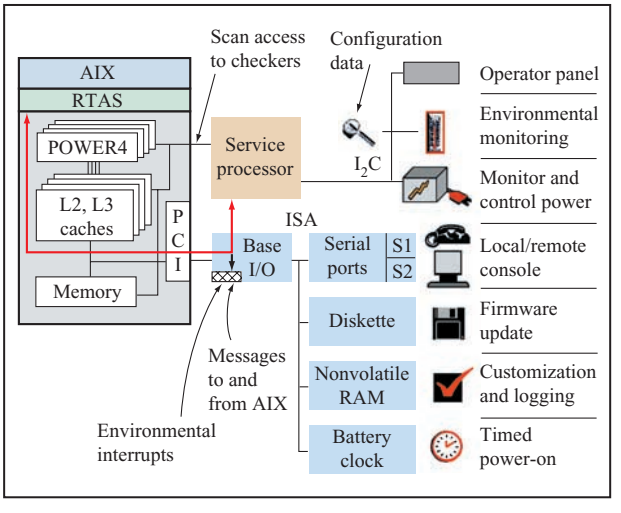

Figure 6

Service processor schematic.

The service processor (**Figure 6**) is a separate independent processor that provides hardware initialization during system IPL, operation monitoring of environmental and

error events, and maintenance support for the p690. For run-time diagnostic purposes, the communication between the service processor and the p690 consists of 1) attention signals from the p690 hardware and 2) read/write communication between the service processor and all hardware-internal FIRs, using specialized JTAG[1] ports between the service processor and all p690 chips. This diagnostic read/write capability of hardware error registers is simultaneous, asynchronous, and transparent to any system activity running on the p690. Described another way, these FIRs are known only to the service processor and are not accessible by system software.

When an error is detected in hardware, an appropriate attention signal is generated to the service processor. The ultimate response of the service processor is to read the appropriate FIR, based on analysis of the WOF structure, to examine the active FIR bits, and to post the FRU callout in the p690 nonvolatile RAM. The NVRAM acts as a "mailbox" for the service processor, the system firmware, and the AIX running on the p690. The formatted FRU callout is moved by system firmware to AIX and into the AIX system error log, along with notification about the nature of the event, usually a deferred repair, based on the p690 internal-element availability mode: CPU, L2, L3, memory, or peripheral component interconnect (PCI) adapter offline. Following the analysis of a recovered event, the service processor resets the FIRs so that they can accurately record any future error events.

Run-time diagnostics for PCI adapters

From the above, it is clear that one of the critical co-requisites of run-time diagnostic isolation is run-time access to the internal error state. For the unique p690 hardware, this access is via special service processor data ports or hardware implementation-specific memory-mapped error registers for run-time system firmware access. For industry-standard PCI adapters, this scan-path access is not currently available. The alternative is to use the existing adapter device-driver access to sense information. IBM encourages device drivers to be written so that they respond to any adapter error indication by reading and logging into the AIX error log all sense data from the adapter. The sense data are examined by an AIX function called Diagnostic Error Log Analysis, which creates the appropriate FRU callout. Before deploying any PCI adapter into an AIX release, the adapter specification is examined to be sure that all internal sense data is 1) logged and 2) defined so that Diagnostic Error Log Analysis will call the correct PCI adapter FRU. This behavior is also tested by IBM during maintenance package verification for the adapter.

[1] JTAG (Joint Test Action Group) is an IEEE working group responsible for certain IEEE hardware testing standards.

3. Internal array and memory fault tolerance

Recovery from soft errors

In POWER4 CMOS technology, the ability to tolerate and mask soft errors in all caches and arrays is paramount. If these errors were not recovered, unacceptable customer outages would result because of aggregate soft-error rates (SERs) on the SRAMs. These SERs exceed by as much as a factor of 5000 the technology-conventional intrinsic failure rate. Error correction at all cache levels and in most large arrays is provided. In the POWER4 L2 and L3 caches, this is accomplished by standard single-error-correct, double-error-detect Hamming ECC [4]. In the L1 data cache (D-cache), because of its store-through design, the L1 D-cache and D-cache tag are parity-protected. Errors encountered are reported as synchronous machine-check interrupts. To support error recovery, the p690 machine-check interrupt handler is implemented in system-specific firmware code. When the interrupt occurs, the firmware saves the processor-architected states and examines the processor registers to determine the recovery and error status. If the interrupt is recoverable, the system firmware removes the error by invalidating the D-cache and incrementing the error counter. If the D-cache error count is greater than a predefined threshold, which is an indication of a solid error, the system firmware disables the failing portion of the D-cache. The system firmware then restores the processor-architected states and "calls back" the operating system machine-check handler with the "fully recovered" status. The operating system checks the return status from firmware and resumes execution. With the D-cache invalidated, data now loads from the L2. Similar recovery function is provided for data ERAT (effective-to-real address translation) and TLB (translation lookaside buffer) arrays.

Figure 7 shows the p690 array-recovery flow chart. For the I-cache, I-cache tag, and I-ERAT, the hardware reports parity errors as if a cache miss had occurred, causing a refetch from an L2 or I-ERAT miss and a reload from the TLB. In the L2 ECC implementation, correct data is always written back into the L2, and the correct data is actually refetched from the L2 cache. This is different from most main-store ECC implementations, where corrected data is presented to the fetch requester but not rewritten into the main store until a special write operation such as scrubbing takes place.

Redundancy for array availability

While the most likely failure event in a processor is a soft single-bit error in one of its caches, other events can occur

and must be distinguished from one another. For the L1, L2, and L3 caches and their directories, hardware and firmware keep track of whether permanent errors are being corrected beyond a threshold. If the threshold is exceeded, a deferred-repair error log is created. Additional run-time availability actions, such as CPU vary off or L3 cache-line delete, are also initiated. L1 and L2 caches and L2 and L3 directories on the POWER4 chip are manufactured with spare bits in their arrays, which can be accessed via programmable steering logic to replace faulty bits in the respective arrays. This is analogous to the redundant bit steering, employed in main store as a mechanism to avoid physical repair, that is also implemented in POWER4 systems. The steering logic is activated during processor initialization. It is initiated by the built-in self-test (BIST) at power-on time. L3 cache redundancy is implemented at the cache-line granularity level. Exceeding correctable-error thresholds while running causes the invocation of a dynamic L3 cache-line delete function, capable of up to two deletes per cache. In the rare event of solid-bit errors exceeding this quantity, the cache continues to run, but a message calling for deferred repair is issued. If the POWER4 system is rebooted without such repair, the L3 cache is placed in bypass mode, and the system comes up with this cache deconfigured.

Main-memory fault tolerance

The problem of detecting and correcting multiple errors originating from a single point of failure in memory was first described from the standpoint of detection [5], using standard Hamming ECC. The solution to this problem implemented in p690 main memory is called *chip-kill ECC*, and it refers to designing the memory-card wiring so that every system ECC word has all of its individual bits mapped onto different DRAM chips. The effect is that not only are single-cell failures corrected on the fly, but also a more serious defect such as an entire DRAM chip kill can be corrected on the fly, since even such a large number of bad bits still appear to the ECC logic as correctable single-bit errors.

4. System recovery from data corruption

POWER4 synchronous machine-check interrupt

In order to support non-checkstop behavior in the unlikely event of corrupt data from a variety of sources, the POWER4 CPU core is designed to respond to load data which has either a parity error from L1/ERAT/TLB/SLB or an uncorrectable error from L2 with a synchronous machine-check interrupt. The p690 system-specific firmware machine-check interrupt handler examines the POWER4 machine and error state and determines the appropriate response. In the case of L1, TLB, or ERAT

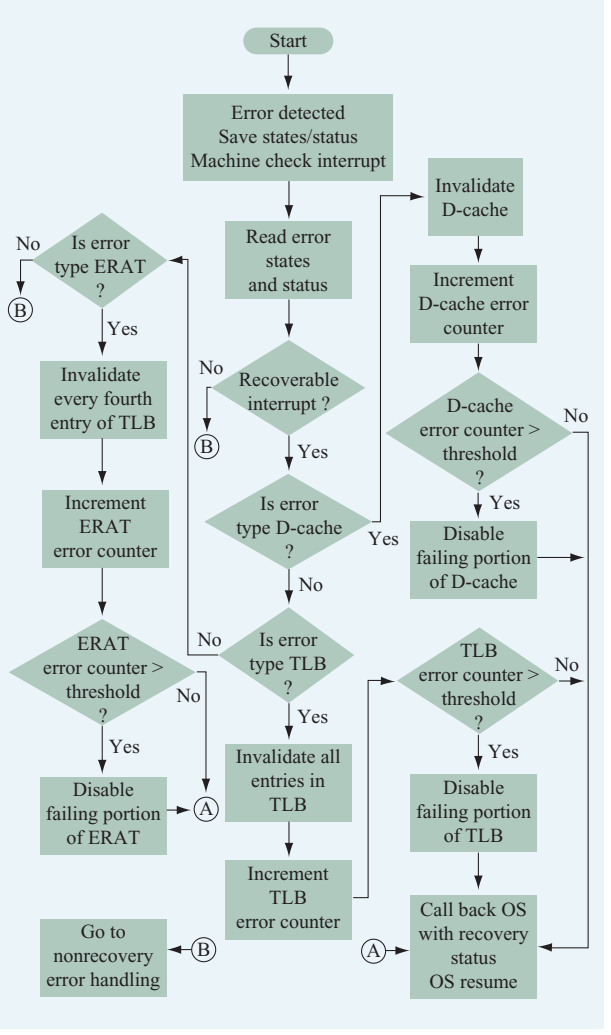

Figure 7

Array error recovery flowchart for the p690.

parity error, the firmware initiates recovery as described in the previous section. For the case of an uncorrectable error, the firmware machine-check handler restores the processor-architected states so that the SRR0 register contains the effective address of the instruction which referred to the corrupt data at the time of interrupt. The firmware machine-check handler then "calls back" the operating system machine-check handler with the "not recovered" and "error sync" (error synchronous to the current context) status. Operating-system recovery consists of determining the nature of the affected process and initiating user process termination, kernel process restart, or operating system termination, depending on whether the kernel was damaged. In a logical partitioned (LPAR) system, only the affected partition is terminated, which

localizes the error impact and maintains overall p690 system availability. The affected partition can be restarted with a fast partition reboot.

System recovery hierarchy

POWER4 system error handling and recovery has multiple layers to maximize system availability. ECC throughout the system and bus retry accomplish recovery for the bulk of the errors expected. In the rare event that the ECC capability is exceeded, or bus retry fails to provide correct data, predecessor machines will respond with some form of system checkstop to maintain data integrity. For POWER4 systems, the hardware, firmware, and AIX are designed to keep running, avoiding a hardware checkstop while maintaining data integrity. Any corrupt data is tagged with a variety of hardware indicators (special unrecoverable-error syndrome in memory, Derr signal on buses, special error packets on I/O transmissions, etc.). System error handling occurs only when corrupt data is referenced. For the case of a processor load, the POWER4 synchronous machine-check interrupt indicates to the operating system the address of the instruction loading the corrupt data. In an LPAR system, only the partition referencing the corrupt data will have an outage. If the kernel is not affected by the data error, only the user process is terminated.

5. PCI fault tolerance

On a large server such as the p690, the total number of included PCI adapters and devices accounts for a significant fraction of the total opportunity for failure, as high as 25% of the total expected outages on a fully configured system. While PCI "hot swap" is supported with both hardware and firmware enablement, PCI hot swap does not deal directly with run-time recovery for an operating adapter. It does allow concurrent repair and continuous operation in configurations where failover to a redundant or alternate path is provided. Run-time recovery from PCI failures has two significant components based on the origin of the error or failure: 1) unique internal device recovery and error status reporting combined with AIX device-driver behavior in the event of a fatal error, and 2) the ability to recover from errors originating on the PCI bus itself. The first category of recovery has been standard in AIX device drivers and is well understood. The second category, errors originating on the PCI bus itself, is not covered by industry-standard PCI device drivers, and the effect has been system global machine-check interrupts, usually not recoverable.

Beginning with the p690, new PCI bus hardware behavior is combined with system firmware and new AIX device-driver behavior to support command-level retry recovery from PCI bus intermittent errors, and device-driver-supported system recovery for permanent PCI bus errors, including partition error containment for LPAR.

"Extended error handling for PCI" is the term used to describe the combination of hardware, firmware, and device-driver behavior which accomplishes the recovery. The first requirement is a hardware mechanism that can report to a software device driver that a PCI bus error has been encountered. The p690 PCI bridge hardware has been designed so that PCI bus parity errors cause the bridge logic generating the PCI bus to enter a freeze state and return a special data pattern to the requesting device driver. The device driver that receives this data pattern then uses special firmware functions to interrogate the specific PCI bridge internal error registers to determine the cause, and can invoke a firmware function to unfreeze the bridge and retry the operation. The specific fault-isolation state is also captured during this activity, and even if it is successful, error thresholds are kept and analyzed to call out the PCI adapter and slot if the threshold is exceeded. On the basis of failure statistics of pin connectors used in standard PCI adapters, PCI bus error recovery can account for reducing as many as 250 events per 1000 systems per year; it is a critical fault-masking technique for achieving the p690 goal.

6. Minimum-impact availability modes

Internal redundancy within the p690 cache arrays has already been described, and is the first line of defense for permanent errors, even though ECC continues to correct them. These mechanisms provide full availability without degradation, and do not require even a deferred repair action. This was described in Section 3. Beyond this capability, for failures which may exceed the capacity of the internal redundancy, individual small elements within the p690 system can be completely deconfigured to allow continued system operation with minimum performance impact.

Deconfiguring a failed element

The p690 is designed to provide the maximum possible computing power, even when it is necessary to deconfigure a failing CPU, cache, or memory for deferred repair. The hardware and firmware are designed to be able to deconfigure the minimum functional element either dynamically during system run time or during boot time, with the remainder of the system unaffected. A p690 processor or an L3 cache line can be dynamically deconfigured while the system is running, and a processor, memory segment, or L3 cache module can be persistently deconfigured during boot time to avoid rediscovery of a failure while awaiting a repair. In the case of the L3 cache, of which there are 16 in a 32-way p690, each individual L3 module can be placed by firmware in a

bypass mode, with all 32 CPUs, all of main memory, and all of the remaining 15 L3 modules forming an operational p690. For the case of an L2 cache ever becoming inoperative, it is necessary to deconfigure the two CPUs which operate out of it. Individual memory cards of the p690 can be deconfigured for deferred repair, and the follow-on system will allow deconfiguration of a half-card, with the remaining half-card fully addressable.

7. Verification of fault-tolerant behavior

Verification of run-time self-diagnostic fault isolation

There are several hardware, firmware, and software behaviors that contribute sequentially to the end-to-end diagnostic behavior, and each is tested and verified independently during its phase of development, as well as being tested and verified as a complete behavior. During hardware chip development, the critical behavior is error detection within the individual error domains, the correct setting in logic of the FIR registers, and the driving of attention-notification signals to the service processor. Prior to chip final design, these behaviors are simulated using logic-model error injection.

Prior to the first-pass release of any chip logic design, the processor run-time diagnostic team conducts detailed design walkthroughs of the chip hardware specification, with specific focus on the FIR design, to make sure that FRU callouts are correctly identified and documented in the chip specification, even though the actual diagnostic code will not be written for about two years.

The final phase of verification is to perform run-time error injection on production-level hardware, firmware, and software. To enable this activity, special error-inject functions are designed into the POWER4 chip set. These hardware functions are programmed by specialized firmware running in the service processor, which can scan special patterns into any one of the hundreds of error-checker domains, causing a single-cycle "bit flip" to occur in either a parity-protected or ECC-protected data packet. The test engineer injects a large number of specific individual errors, each one with a predetermined outcome as to recovery, availability mode, threshold, and FRU callout. Incorrect behavior of any kind is documented for a design change. The inject points are chosen throughout the CPU, cache, memory, system bus, PCI, and power and cooling subsystems.

Verification of recovery and availability modes

The basic foundation of all recovery and availability modes is the run-time fault-isolation behavior, and the basic verification activities described above cover the bulk of these. Areas such as buses, where retry is the recovery mode, are verified using the same error-inject methodology, but the test consists of verifying the fact that retry actually occurred and was successful, and that the error state captured indicated the specific bus and its recovered error count.

There are several availability modes that are invoked only when thresholds of recovered errors are exceeded, leading to an isolated component being varied offline. To verify this behavior, the error-inject logic has a second mode in which it can be programmed to continuously inject a stream of errors at a fixed interval. This allows testing of any such threshold behavior and identifies the subsequent correct availability mode.

Maintenance package verification

This activity extends run-time diagnostic verification down to the more classical "solid bug" error behavior. This allows validation that the error-recreating POST, ABIST, LBIST, and wire-test activities that occur during the IPL sequence can find and isolate electrical opens and shorts, as well as those internal problems that such pattern testing can find. More significant in this phase of verification is that the full production-level service documentation is included, service notification to an external location can be tested, actual field service personnel conduct parts of the test, and procedures involving physical removal and replacement can be tested.

8. Summary

The IBM eServer p690 system contains many significant design improvements in fault tolerance over previous systems, all of which are based on run-time self-diagnostic technology as the fundamental building block throughout the hardware system design: CPU, cache, memory, I/O, and power and cooling. This capability permits a wide variety of concurrent and deferred repairs to take place for all system elements. In addition, internal ECC, spare bits, and predictive deconfiguration availability modes are provided to keep the p690 operating. Self-diagnosis as opposed to manual diagnosis enables automatic dispatch of the correct repair part simultaneously with the placement of the service call, even before the service person arrives on the scene.

Acknowledgments

Many designers have made significant contributions to p690 fault-tolerant design, and it is not possible to mention all of their names. The authors wish to acknowledge the contributions of several people from different technical disciplines in IBM who have provided technical leadership in their areas of the fault-tolerant design of the p690: E. J. Silha, R. L. Arndt, S. M. Thurber, D. O. Lewis, and R. L. Hicks.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of The Open Group.

References

1. Nandakumar N. Tendolkar and Robert L. Swann, "Automated Diagnostic Methodology for the IBM 3081 Processor Complex," *IBM J. Res. & Dev.* **26,** 78–88 (January 1982).
2. C. L. Chen, N. N. Tendolkar, A. J. Sutton, M. Y. Hsiao, and D. C. Bossen, "Fault-Tolerance Design of the IBM Enterprise System/9000 Type 9021 Processors," *IBM J. Res. & Dev.* **36,** 765–779 (July 1992).
3. D. C. Bossen and M. Y. Hsiao, "Model for Transient and Permanent Error-Detection and Fault-Isolation Coverage," *IBM J. Res. & Dev.* **26,** 67–77 (January 1982).
4. C. L. Chen and M. Y. Hsiao, "Error-Correcting Codes for Semiconductor Memory Applications: A State-of-the-Art Review," *IBM J. Res. & Dev.* **28,** 124–134 (March 1984).
5. D. C. Bossen, L. C. Chang, and C. L. Chen, "Measurement and Generation of Error Correcting Codes for Package Failures," *IEEE Trans. Computers* **C-27,** 203–207 (March 1978).

Received June 1, 2001; accepted for publication September 1, 2001

Douglas C. Bossen *IBM Server Group, 11400 Burnet Road, Austin, Texas 78758 (dbossen@us.ibm.com).*

Dr. Bossen is a Distinguished Engineer specializing in computer hardware fault-tolerant design. He has designed and holds numerous patents on error-correcting codes, error-detecting logic, error-detection/fault-isolation techniques, and system availability techniques which are implemented in IBM systems 308X, 3090, ES/9000, and pSeries servers. Since joining IBM he has received two IBM Outstanding Innovation Awards and an IBM Corporate Award for his work on error detection and fault isolation. His current position in Server Group RAS Architecture requires focus on Server Group RAS competitive leadership by migrating the most cost-effective techniques within IBM's extensive portfolio into the pSeries and iSeries brands during product development. He has published 18 peer-reviewed journal articles, holds 24 issued and nine pending U.S. patents and 23 published disclosures, and has reached IBM's Tenth Invention Plateau. Dr. Bossen received the B.S., M.S., and Ph.D. degrees in electrical engineering, all from Northwestern University. He is a Fellow of the Institute of Electrical and Electronics Engineers and a member of the IBM Academy of Technology.

Alongkorn Kitamorn *IBM Server Group, 11400 Burnet Road, Austin, Texas 78758 (kitamorn@us.ibm.com).*

Mr. Kitamorn is a Senior Engineer specializing in system RAS architecture and design. He has designed and holds numerous patents on system availability and error-handling design which are implemented in IBM RS/6000 and pSeries servers. His current position in Server Group Engineering Software Architecture requires focus on architecture, design, and implementation of leading-edge RAS functions in iSeries and pSeries servers. Since joining IBM, he has received numerous Invention Achievement Awards and an Outstanding Technical Achievement Award. He has five issued and 14 pending U.S. patents and three published disclosures, and has reached IBM's Fourth Invention Plateau. He received the B.S. degree in computer science from Union College.

Kevin F. Reick *IBM Server Group, 11400 Burnet Road, Austin, Texas 78758 (reick@us.ibm.com).*

Mr. Reick is a Senior Technical Staff Member specializing in all aspects of system RAS and system debug design. Through his work on IBM S/390 machines, power-parallel nodes, and RS/6000 and AS/400 servers, and by his involvement in the introduction of numerous successful products, he has extensive experience in systems, storage, service processor and microprocessor architecture, architecture RAS design, system debug, and design for test. He is currently the POWER4 RAS Architect and had a lead role in POWER4 system bringup. He has two issued and 26 pending U.S. patents and four published disclosures, and has reached IBM's Fifth Invention Plateau.

Michael S. Floyd *IBM Server Group, 11400 Burnet Road, Austin, Texas 78758 (mfloyd@us.ibm.com).*

Mr. Floyd is a Staff Engineer specializing in RAS and debug design of microprocessors and systems. He has developed a "design for debug" approach and methodology to chip design for the IBM POWER family of servers. Mr. Floyd holds a master's degree in electrical engineering from Stanford University, with a focus on computer hardware design, test, and fault tolerance, and he received a bachelor's degree in computer engineering from the Georgia Institute of Technology. His nearly seven years of experience with IBM includes bringup, debug, and test of the PowerPC 620 microprocessor and POWER4 microprocessor and systems (for iSeries and pSeries). He is currently the POWER5 Microprocessor Core RAS design lead. Mr. Floyd has issued one U.S. patent with more than twenty pending and has reached IBM's Fifth Invention Plateau in addition to co-authoring two published papers.