Motivating Commodity Multi-Core Processor Design for System-level Error Protection

Nidhi Aggarwal, Parthasarathy Ranganathan, Norman P. Jouppi,
James E. Smith, Kewal K. Saluja, and George Krejci

Abstract—**In this paper we analyze the reliability and availability features in several commodity chip multiprocessors (CMPs) and find that they have numerous single points of failure. Failures in some system components, e.g., interconnect, cache controller and memory controller logic, leave CMPs susceptible to error even if the computation is dual modular redundant (DMR) or triple modular redundant (TMR). Furthermore, even though some replicated resources are present in CMPs, they can not be used effectively for providing system-level protection because of the lack of fault isolation in shared components. We describe a CMP design that can provide system-level error protection. The proposed design provides mode configuration features in hardware to tolerate errors in any component. The proposed CMP has two modes: a high performance mode where most of the resources are applied towards performance and a high-availability mode that uses resources, in a redundant manner, to provide tolerance against errors. Our design requires minimal hardware changes and can retain the commodity economics and performance advantages of current CMPs.**

***Index Terms*— Availability, Computer fault tolerance, Reliability, System Recovery.**

I. Introduction

Technology scaling is likely to make future processors more susceptible to hardware errors. Permanent or intermittent hardware faults, caused by defects in the silicon or metallization of process package and wear out over time, lead to "hard faults". Transient faults (or "soft errors"), which cause random bit values to change erroneously, may be caused by electrical noise (e.g., crosstalk) or external radiation (e.g., alpha radiation from impurities). Recent studies [1] [2] [3] have forecast between a two to nine-orders-of-magnitude increase in logic circuits' soft error rates.

Continued technology scaling and diminishing uniprocessor power efficiency have also led to the emergence of Chip Multiprocessors (CMPs) as the predominant hardware paradigm. Recent and proposed microprocessor chip designs from all the key vendors (Intel, IBM, SUN, and AMD) are CMPs. One of the main benefits of CMPs is that on-chip components can be easily shared to improve resource utilization. Common examples are multi-threaded cores, shared last level caches, and shared I/O interfaces. The Sun Niagara includes eight multi-threaded cores, with a shared second-level cache, and integrated memory controllers and I/O interfaces [4]. Future designs are likely to have even more cores on a single chip with greater levels of system integration and resource sharing than in previous generations (e.g., Intel's demonstration of an 80-core processor at Intel Development Forum, 2006).

Increased integration and sharing in a CMP exacerbates the soft error problem, however, due to the additive effect of component FIT rates and potentially larger impact from a single error. A typical solution for handling soft errors in high availability systems is to replicate the computation and compare the results to detect an error [16, 29] and then do either backward or forward error recovery [30]. This method is based on the key assumption that the redundant computation threads are "fault-isolated" and a single soft error does not affect multiple redundant threads. However, shared resources in CMPs pose a problem because they may violate this assumption unless special steps are taken.

In this paper, we examine the fault isolation problem in a CMP for an illustrative high availability system architecture similar to the NonStop Advanced Architecture [16]. The first part of the paper characterizes recent CMP designs from key vendors and shows that current commodity CMP designs do not satisfy the "fault isolation" assumption and are susceptible to errors in certain system components – interconnect, cache controller and memory controller logic – even if a computation is performed in a dual modular redundant (DMR) or triple modular redundant (TMR) mode. The point is that even though replicated cores are present in a single CMP, they are not sufficient for providing system level fault protection because of the lack of fault isolation in their shared components. For example, even if the cores in Intel Xeon MP [8] are used for redundant computation, a transient fault in the shared interface to the front side bus or the L2 cache bank controller logic can cause an undetected error.

In the second part of the paper, we describe ongoing work on a new architecture that allows for configurable isolation to create "fault zones" with strict isolation properties that software can then intelligently allocate in order to satisfy availability requirements. The proposed CMP has two modes. A high performance mode applies most of the resources toward performance and can be used by applications that can tolerate soft errors. A high-availability mode uses resources in

Manuscript submitted January 2, 2007. This work was done at HP Labs, Palo Alto, California. N. Aggarwal, J. E. Smith, and K.K. Saluja are with University of Wisconsin-Madison, Madison, WI 53715 USA (e-mail: naggarwal@wisc.edu, jes@ece.wisc.edu, saluja@engr.wisc.edu).

P. Ranganathan, N. P. Jouppi, and G. Krejci are with HP, Palo Alto, CA 94304 USA (email: partha.ranganathan@hp.com, norm.jouppi@hp.com, george.krejci@hp.com)

a redundant manner to provide tolerance against soft errors. "On demand" configurable isolation enables a high volume processor to provide high soft error coverage for mission critical applications with very low cost. Mode reconfiguration takes place at boot time in most systems, although other alternatives are possible. Our proposed design requires minimal hardware changes and retains the commodity economics and performance advantages of current CMPs. As an added benefit, configurable isolation can also be used to reconfigure the system for graceful degradation in the event of hard faults or for product binning during testing.

II. Reliability features in commodity CMPs

We analyzed the reliability and availability features of five commodity CMP architectures from key vendors – AMD Opteron [8], SUN Niagara [4], Intel Xeon [10], Intel Montecito [11] and IBM POWER [12] [13]. We divided each of the CMPs into different components and then characterized whether they satisfied key requirements for fault tolerance as specified by White et al. [14]: redundancy, fault isolation, fault detection and online repair. For brevity, we present a brief summary of the analysis here.

Cores – Inside the core, transient fault detection is mainly restricted to register files. An exception is Montecito where there is built-in lockstep support with internal soft error checking capabilities. There is no fault isolation in Opteron, Xeon and Niagara; an error originating in any core can propagate to all the other cores through the shared system components. Power5 and Montecito provide some degree of isolation for cores in different logical or electrical partitions, respectively. In summary, all the commodity CMP architectures are vulnerable to soft errors, except Montecito in its lockstep configuration.

Caches – Most architectures are resilient to errors in the cache array and provide ECC or parity checking at all cache levels. However, Opteron and Xeon can not tolerate errors that are not correctable by ECC alone, for example multi-bit errors. Niagara, Power5 and Montecito have more redundancy and fault isolation and can tolerate important classes of multi-bit errors. These CMPs share at least one level of the cache hierarchy either across cores or contexts. However, none of the commodity CMPs can tolerate errors in the associated cache circuitry or interconnect. For example, if all L2 cache banks are shared, and addresses are interleaved among the banks, a transient failure in the cache controller state machine could lead to erroneous setting of a coherence bit. Note that ECC on the coherence state bit would not prevent this error because the fault is in the cache controller logic and not the actual coherence bit. Such an error could affect an entire socket.

Memory – Memory is perhaps the most fault tolerant resource in commodity CMP systems. All the conditions for fault tolerance are satisfied in the memory arrays. This also reflects the fact that historically memory is the most error prone component in a system. All the architectures have sophisticated techniques like chip kill, background scrubbing, and DIMM sparing to tolerate failures. However, there is no tolerance to failures in memory access control circuitry. A failure in any memory controller or anywhere in the interconnect would affect all the cores. For example, in a design like the Xeon, an error in one memory controller in the shared NorthBridge can affect multiple cores. On the other hand, in Opteron the failure of an on-chip memory controller can be potentially isolated to the cores in that chip.

In summary, we found that existing transient fault detection is limited to storage arrays such as register files, cache and memory arrays. Also, the lack of system level fault isolation is the biggest problem. Shared components do not have adequate fault isolation because a fault in one of the shared components can affect all the cores on the chip. This is true even if programs are being run in a Dual Modular Redundant (DMR) or Triple Modular Redundant (TMR) configuration. Instead it is preferable to have architectures where the effects of faults can be isolated to individual threads. CMP designs should provide the ability to tradeoff high error coverage with high performance.

III. Proposed CMP design

In this section we describe a CMP design with features for providing system-level soft error protection. These features allow the system to be configured into two modes: a high performance mode and a high availability mode. In the high performance mode, the CMP resources are shared for maximum utilization. In high availability mode, the degree of resource sharing is configured to provide selective isolation. This allows the resources to be used in a redundant manner and to prevent correlated errors. Mode reconfiguration can take place at boot time in most systems, though other alternatives using ACPI [29] are possible.

Although the proposed method can be implemented in various CMP designs, we use one representative system as an illustrative example. Figure 1(a) shows a conventional CMP architecture with 8 cores (P0...P7) and private level-1 caches, an 8-way banked shared level-2 cache, 4 memory controllers, and coherent links (such as Hypertransport [8]) to other sockets or I/O hubs. For this discussion we assume a bidirectional ring interconnect between cores and banks. (The proposed techniques can also apply to more complex 2-D arrangements such as meshes). Although a classic "dance hall" layout is shown for simplicity (with all the cores on one side and the shared cache banks on the other side), designs with banks and cores interleaved are also within the scope of the proposed method.

A very straightforward approach is a design with a number of fully independent microprocessors fabricated on the same die (Figure 1(b)). Each has its own memory controller and I/O connections. For high availability, sets of independent cores are configured into a DMR or TMR mode. However this architecture has a number of disadvantages. Hard partitioning of cache resources (inhibiting any sharing) when

in the high performance mode significantly reduces overall system performance [5], and this method has not been used in proposed CMP designs [4, 33, 34]. Similarly, valuable pin bandwidth is inefficiently used. This makes such a design unattractive for high volume applications that have lower availability requirements, and it reduces the independent-microprocessor approach to a niche design that would be economically unfeasible.

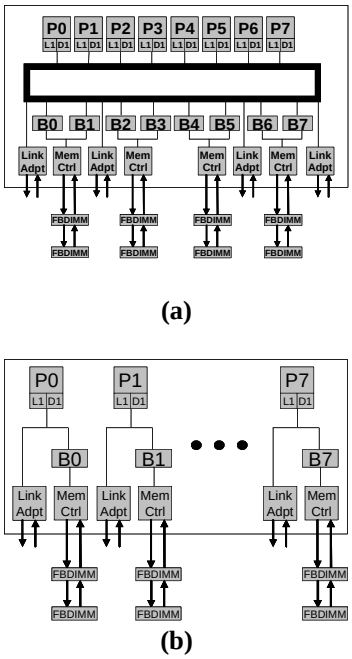


Figure 1. CMP architecture designs. (a) Baseline conventional system architecture and (b) A system with full isolation.

We propose the addition of simple hardware to the baseline CMP to enable in-field boot time configuration of the system. This provides the combination of enhanced fault isolation in a high availability configuration and negligible performance loss when in a high performance, non-fault tolerant configuration. A block diagram of the proposed organization is shown in Figure 2. The components are "color coded" where color domains act as units of fault containment. Hardware is configured to limit errors caused by a fault to a single color domain. Any error in a color-shared component affects computation only on the cores mapped to that color and prevents correlated errors that are undetectable by voting with a redundant computation in another color. To ensure that an error in one color domain does not propagate to all the other color domains on the CMP, we propose "configurable isolation" for interconnect, caches and memory controllers. We define configurable isolation as *techniques that allow the system to be configured with different levels of isolation by controlling resource sharing.*

The proposed architecture offers various capabilities. The number of colors (isolation domains) is not restricted to two and can be increased either to provide more redundancy or smaller granularity of fault containment. For example, three colors can be used with a voter in a Triple Modular Redundant (TMR) configuration. Furthermore, the number of colors need not be static, and if the level of protection required changes then the number of colors can be reconfigured by system software. The proposed approach is suitable for integration both in Backward Error Recovery systems (in a DMR configuration in concert with check pointing solutions similar to Revive [19], SafetyNet [20] or [21]), or Forward Error Recovery systems (in a TMR configuration). The proposed design not only provides fault isolation without losing the benefits of sharing, it also offers the ability to reconfigure in the event of hard faults for graceful degradation.

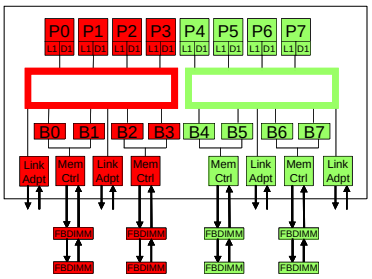


Figure 2. Proposed architecture (two color configuration)

Below, we consider the application of the proposed method to implement a NonStop-like DMR [16] configuration as an illustrative example. Resources from two colors define a DMR process pair, with computations in the red domain replicated in the green domain when higher availability is required. The microprocessor can be reconfigured to the higher-availability configuration by setting a small number of control points in the design. This allows systems to support "availability on demand" in-field.

Note that this capability only requires small changes to the ring and bank addressing, while the rest of the CMP is unchanged. The ring interconnect in Figure 2 has been cut apart and reconfigured to create two logically independent rings by activating cross links. Because the ring is expected to be placed down the center of the chip, the cross links should be less than a millimeter long, and their activation requires the insertion of a multiplexer at the input of a ring interface incoming data port. Thus cross links and input multiplexers are a small additional fixed cost in terms of area and power, which does not significantly increase the cost of the design in cases were higher availability is not desired. The cross-links are also expected to be shorter than the ring segments between cores. Thus the cross-connects should be able to operate at least as fast as a core to core or bank to bank ring segment. In combination with partitioning of the inter-core interconnect, in high availability mode the interleave among level 2 cache banks uses one fewer bit, interleaving references among half the banks, keeping references within the same color.

In high availability configurations, voters that compare the redundant computation can be implemented in a number of different ways. For highest availability, voters can be

implemented in I/O hubs connected to a red and green link adapter, similar to the hardware voters in the Nonstop Advanced Architecture [16]. For lower-cost lower availability solutions, the voter can be implemented in software hypervisors that communicate between the colored partitions through the I/O system [18]. Similar to the NonStop system, physical memory can be partitioned between the logical processors using unique virtual to physical memory mapping.

IV. Evaluation and Results

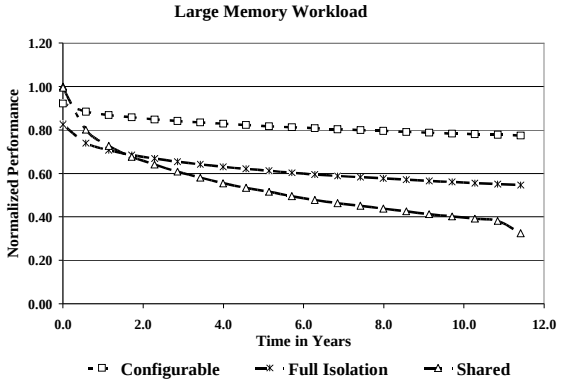
We evaluated the impact of hard faults and subsequent reconfiguration on the computing capacity of the system over its lifetime. We compare three architectures: 1) **shared** -- a completely shared system similar to proposed CMPs (Figure 1a), 2) **fully isolated** -- a completely private system with full isolation (Figure 1b), and 3) **configurable isolation** -- the proposed architecture with reconfiguration and configurable isolation (Figure 2). All three are assumed to be in a DMR configuration.

Because the proposed architecture does not contain any modification to the cores, the most important workload characteristic is the size of the working set and its effect on cache behavior. We constructed three workloads with large, mixed, and small memory footprints using SPEC benchmarks. Over the course of a simulation run, as cores become unusable due to hard faults, benchmarks are dropped from the workloads, reflecting the loss of computing capability.

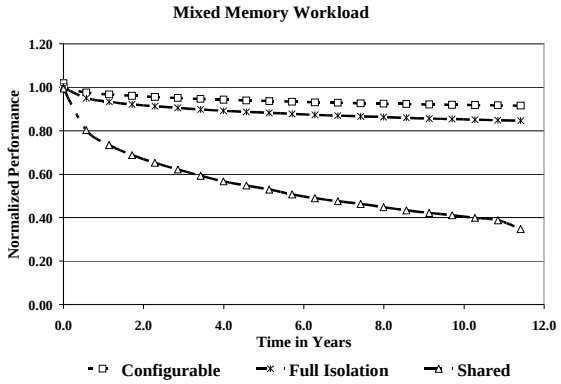
The fault model is based on state of the art technology and is derived from detailed (confidential) models from processor vendors. The fault model was calibrated by HP-internal fault analysis experiments. The fault data includes failures in time (FIT) rates and distributions for hard and soft errors per component. The system is divided into five different fault zones (that represent the granularity of reconfigurations): core and L1 cache, L2 circuitry, L2 banks, memory controller circuitry, and link controller.

On the shared system any hard fault leads to system failure. This means that after a failure, the throughput of such a system goes to zero for all workloads. On the fully isolated system any single fault leads to the loss of throughput from a process pair in a DMR configuration. For example, even a fault in the bank associated with a core leads to that core being unusable. On a configurable isolated system, a reconfigurable fault (for example, in a memory controller) leads to loss of performance over all the process pairs, but not the loss of a workload. Only when a core fails, a benchmark (both copies) is dropped from the workload. The entire system becomes unusable in the configurable isolated architecture only when the penultimate component of any type fails (for example, 7th core, 3rd memory controller, 7th cache bank in a system similar to the baseline).

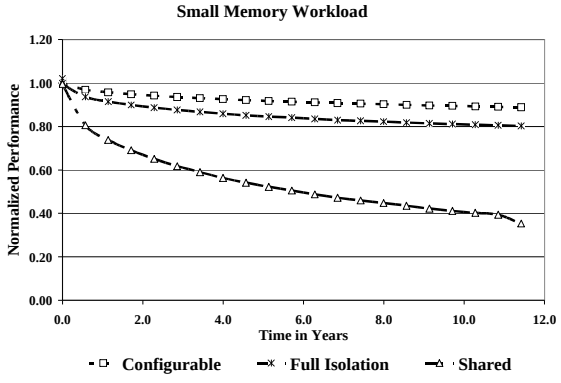
To make evaluation feasible, we use a two phase simulation methodology for simulating the performance of different processor configurations for various fault arrival scenarios. First, we use a full system simulator to exhaustively simulate the possible system configurations (using more than one machine year) and compute the throughput of all configurations (subject to policies described below). Second, we perform Monte Carlo simulation using a detailed component-level fault model. By running the Monte Carlo simulation for 10,000 runs we simulate fault injection in a total of 10,000 systems with each run comprising 100,000 simulated hours (approximately 11 years).



(a)



(b)



(c)

Figure 3. Normalized performance from Monte-Carlo hard fault simulation over an 11-year period of time. Results for three architectures – a baseline conventional system with full sharing, the proposed system, with configurable isolation, and a system with full isolation. Evaluated workloads include a) large memory workload, b) mixed memory workload and c) small memory workload.

All simulations were done using a full system x86/x86-64 simulator (based on AMD SimNow™) which can boot an unmodified Windows or Linux OS and execute complex application programs. More details about the simulator are in Falcon et al [31]. We use a timing model with a memory hierarchy similar to that supported by an AMD Opteron 280 processor, except with smaller L2 cache sizes to match the working set of the workloads. More details regarding the simulation methodology can be found in Aggarwal et al [32].

Figures 3(a), (b), and (c) summarize the baseline results for the large, mixed, and small memory workloads, respectively. As expected, the shared system performs the worst, with a dramatic degradation in average performance (30-35%) during the first two years, and degradation close to 50% by the end of five years. The fully-isolated configuration is much more resilient to failures and provides more gradual performance degradation. Over five years, the net performance loss is only 10-15%. The results for the large memory workload (Figure 3a) are particularly interesting. Here, the completely isolated configurations, by virtue of having private caches, initially under-perform the shared configuration. However, compared to the fully-shared system, the fully-isolated system becomes performance competitive at around 2 years (at the crossover point in the curves in Figure 3a).

The configurable isolation system consistently achieves the best performance across all the workloads. With configurable isolation, resources can still be shared within a given fault zone. Additionally, the ability to dynamically repartition the resources leads to the best graceful degradation across all three workloads.

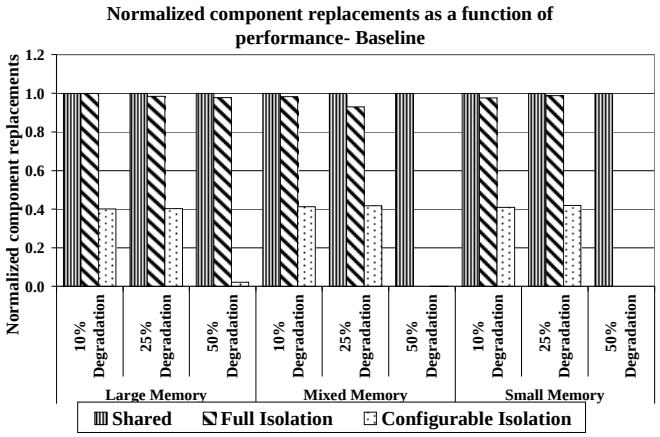


Figure 4. Number of normalized component replacements as a function of performance. The three architectures (mixed workload) are compared, assuming components are replaced (a) when performance dips below 90%, (b) when performance dips below 75%, and (c) when performance dips below 50%.

Figure 4 provides an alternate view of the benefits of configurable isolation. For each of the three approaches, the number of component replacements is shown. It is assumed that the system continues to stay operational until the performance dips below a certain threshold after which the entire multi-core component is replaced (and the performance is re-initialized to that of the no-fault configuration). The simulation then continues for the remainder of the 100,000 hours with the new system. We consider three cases where the performance threshold is set to (a) 90% (b) 75%, and (c) 50% of initial performance. The total number of replacements (across 10000 Monte Carlo runs) for fully isolated and configurable isolated system is normalized with respect to the total number of replacements for a fully shared system. In a fully shared system every fault leads to system replacement because the performance drops to 0. These results show that the architecture with configurable isolation dramatically reduces the need to replace components across all three workloads irrespective of the performance thresholds.

V. Future Work

There are several enhancements to the configurable isolation architecture that provide additional benefits. For example, as discussed earlier, we assumed a single process per core. Some of the performance degradation from losing a component in a given fault zone can potentially be mitigated by overloading processes on remaining components in that fault zone. Similarly, when remapping fault zones, we assume arbitrary remapping of the fault zones and assignment of processes to cores. One can conceive of using more advanced policies that are aware of workload requirements and latency effects to improve performance. For example, prior work on heterogeneous multi-core architectures has demonstrated significant benefits from intelligently mapping workloads to available hardware resources.

An interesting advantage of configurable isolation is the ability to reconfigure dynamically the system availability guarantees. The proposed approach allows the system to be configured to a spectrum of choices from no fault isolation to multiple smaller fault zones. For example, in utility-computing environments, a server can be provisioned as a payroll server with high levels of availability turned on, and then later can be redeployed as a web server with lower levels of availability. Additionally, although we have focused primarily on isolation advantages of the proposed architecture from a fault perspective, equally important are benefits from a performance and security point of view. For example, in cases where a high-priority workload and a low-priority workload both compete for shared resources (e.g., the L2 cache), the proposed architecture can provide performance isolation, to ensure quality-of-service guarantees. Furthermore, this architecture can enable tradeoffs between availability and performance, which is a useful characteristic in utility computing environments.

VI. Related Work

Much of the recent architecture research in fault tolerant systems has focused on tolerating errors originating in the core, for example, DIVA [22], SRT [9], SRTR [23], AR-SMT

[24], CRTR [6], DRM [25], structural duplication [26], banking lifetime [27], TRUSS [7] and several others that use the extra cores or contexts [15] available in a CMP. Most of these require extensive modifications to existing chip designs and/or the addition of new hardware structures. Furthermore, the issues regarding detection and recovery from errors in on-chip components such as bank controller or memory controller logic are not well addressed by solutions that focus on transient failures in the core.

To the best of our knowledge, ours is the first proposal to characterize the impact of logic faults in other components like bank controllers, interconnect and memory controllers at system level in a CMP and present a solution to deal with these faults. Other system level recovery solutions for SMPs like NonStop [16] and zSeries [28] handle errors in the interconnection network and the cache coherence protocol but do not deal with the lack of fault isolation in CMPs.

VII. Conclusions

Future processors are going to be increasingly susceptible to hardware errors. The impact of soft errors on a CMP is likely to be severe because there is a lack of system level fault isolation in the shared resources. We found that current CMPs are susceptible to transient faults even in a DMR or TMR configuration. We argue that all components of the chip must be protected. The goals of a commodity CMP design include the retention of performance advantages while at the same time providing mechanisms to optionally tradeoff high error coverage with high performance.

We describe a CMP design that can provide system level soft error protection and the ability to reconfigure in the event of hard faults for graceful degradation. The proposed design provides configurable hooks in the hardware so that the chip can be configured to tolerate soft errors in any component. Our design requires minimal hardware changes and retains the commodity economics and performance advantages of current CMPs.

Acknowledgment

We are grateful to Paolo Faraboschi for his invaluable help with the simulator. We would also like to thank Prasun Agarwal, Luiz Barroso, Wendy Bartlett, Dave Garcia, Daniel Ortega, John Sontag, Bill Tian, and Shyam Thoziyoor for their input. SimNow™ is an AMD trademark.

References

[1] http://www.itrs.net/Links/2006Update/2006UpdateFinal.htm

[2] Borkar, S., *"Challenges in Reliable System Design in the Presence of Transistor Variability and Degradation"*, IEEE Micro, vol. 25, no. 6, Nov.-Dec. 2005, pp. 10-16

[3] Shivakumar, P. Keckler, S.W., Moore, C.R., Burger, D., "*Exploiting Microarchitectural Redundancy for Defect Tolerance*", the 21st International Conference on Computer Design (ICCD), October, 2003.

[4] Kongetira, P., Aingaran, K., Olukotun, K., *"Niagara: A 32-Way Multithreaded Sparc Processor"*, IEEE Micro, 2005.

[5] Jaleel, A., Mattina, M., Jacob, B., *"Last level cache (LLC) performance of data mining workloads on a CMP - a case study of parallel bioinformatics workloads"*, HPCA 2006.

[6] Gomaa, M. et al., *"Transient-fault recovery for chip multiprocessors"*, ISCA 2003.

[7] Gold, B. T. et al., *"TRUSS: a reliable, scalable server architecture"*, IEEE Micro, Nov-Dec 2005.

[8] Keltcher, C.N., McGrath, K.J., Ahmed, A., and Conway, P., *"The AMD Opteron processor for multiprocessor servers"*, IEEE Micro, 2003.

[9] Reinhardt, S. K. et al., *"Transient fault detection via simultaneous multithreading"*, ISCA 2000

[10] www.intel.com/business/bss/products/server/ras.pdf

[11] McNairy, C., Bhatia, R., *"Montecito: a dual-core, dual-thread Itanium processor"*, IEEE Micro, 2005.

[12] http://www03.ibm.com/systems/p/hardware/whitepapers/power5_ras.html

[13] Bossen, D. C., Kitamorn, A., Reick, K. F. and Floyd, M. S., *"Fault-tolerant design of the IBM pSeries 690 system using POWER4 processor technology"*, IBM Journal of Research and Development, 2002.

[14] White, R.V., Miles, F.M., *"Principles of fault tolerance"*, Applied Power Electronics Conference and Exposition, 1996.

[15] Mukherjee, S. S. et al., *"Detailed design and evaluation of redundant multithreading alternatives"*, ISCA, May 2002.

[16] Bernick, D., Bruckert, B., Vigna, P. D., Garcia, D., Jardine, R., Klecka, J., Smullen, J., *"NonStop® Advanced Architecture"*, DSN, 2005.

[17] Huh, J., Burger, D., Keckler, S.W., *"Exploring the Design Space of Future CMPs"*, PACT', 2001.

[18] Bressoud, T. C. and Schneider, F. B., *"Hypervisor-based fault tolerance"*, ACM Trans. Comput. Syst. 14, 1 (Feb. 1996), 80-107.

[19] Nakano et al., *"ReViveI/O: Efficient Handling of I/O in Highly-Available Rollback-Recovery Servers"*, HPCA 2006.

[20] Sorin, D. J. et al., *"SafetyNet: improving the availability of shared memory multiprocessors with global checkpoint/recovery"*, ISCA, 2002.

[21] Masubuchi, Y. et al., *"Fault recovery mechanism for multiprocessor servers"*, In Proceedings of the 27th International Symposium on Fault-Tolerant Computing, pages 184–193, 1997.

[22] Austin, T. M., *"DIVA: A reliable substrate for deep submicron microarchitecture design"*, MICRO 1999.

[23] Vijaykumar, T. N. et al., *"Transient-fault recovery using simultaneous multithreading"*, ISCA 2002.

[24] Rotenberg, E., *"AR-SMT: A microarchitectural approach to fault tolerance in microprocessors"*, In Proceedings of the 29th International Symposium on Fault-Tolerant Computing, June 1999.

[25] Srinivasan, J., et al., *"The Case for Lifetime Reliability-Aware Microprocessors"*, ISCA 2004.

[26] Srinivasan, J., et al., *"Exploiting Structural Duplication for Lifetime Reliability Enhancement"*, ISCA 2005.

[27] Lu, Z. et al., *"Banking chip lifetime: Opportunities and implementation"*, High Performance Computing Reliability Issues, 2005.

[28] M.L., Fair et al., *"Reliability, Availability, and Serviceability (RAS) of the IBM eServer z990"*, IBM Journal of Research and Development, Nov, 2004.

[29] http://www.acpi.info/

[30] Anderson, T., Lee, A., *"Fault-tolerance - Principles and Practice"*, Prentice Hall, Eaglewood Cliffs, 1981.

[31] Falcon, A. Faraboschi, P., and Ortega, D., *"Combining Simulation and Virtualization through Dynamic Sampling"*. ISPASS-2007.

[32] Aggarwal, N., Ranganathan, P., Jouppi, N. P., and Smith J. E., *"Using configurable isolation to achieve high availability systems using commodity multi-core processors"*, to appear in ISCA 2007.

[33] Barroso, L. A., Gharachorloo, K., McNamara, R., Nowatzyk, A., Qadeer, S., Sano, B., Smith, S., Stets, R., and Verghese, B., *"Piranha: A scalable architecture based on single-chip multiprocessing."* In Proceedings of the 27th International Symposium on Computer Architecture, June 2000.

[34] Tendler, J. M., Dodson, J. S., Fields Jr., J. S., Le, H., and Sinharoy, B., *"IBM Power4 system microarchitecture"*, IBM Journal of Research and Development, 46(1):5–26, 2002.