OpenSPARC: An Open Platform for Hardware Reliability Experimentation

Ishwar Parulkar and Alan Wood, *Sun Microsystems*
James C. Hoe and Babak Falsafi[*], *Carnegie Mellon University*
Sarita V. Adve and Josep Torrellas, *University of Illinois at Urbana-Champaign*
Subhasish Mitra, *Stanford University*

***Abstract*—OpenSPARC is an open source community based around hardware design and experimentation aids for the UltraSPARC^{TM} T1 and T2 chip multi-threaded (CMT) microprocessors[1]. The UltraSPARC^{TM} T2 processor is the industry's first "server on a chip", with 8 cores, 64 threads and on-chip networking and security. The richness of the RTL source code, tools and information in OpenSPARC has made it a comprehensive, practical and relevant platform for research in several areas of computing. This paper highlights the potential of using OpenSPARC for research in hardware reliability. Examples of university research projects, results achieved, benefits gained and lessons learned using OpenSPARC are described. Future research directions in reliability based on OpenSPARC are proposed.**

***Index Terms*—Multi-threaded microprocessors, Open Source, Reliability, Fault Tolerance, On-line Test**

I. Introduction

Historically, microprocessors have been designed to improve the execution performance of single thread programs by exploiting instruction level parallelism (ILP). Common techniques used to improve single thread performance are deep pipelines, multiple instruction issue, speculation, and out-of-order instruction execution. Recently, these techniques have reached a point of diminishing returns because of inherently low or hard to exploit application ILP [2]. Techniques used to improve single thread performance often give rise to complex processor designs with poor pipeline efficiencies and high power consumption

OpenSPARC is based on Sun's UltraSPARC^{TM} T1 [3],[4] and T2 [5],[6],[7] microprocessors, which are designed for commercial workloads that exhibit large amounts of thread level parallelism (TLP). UltraSPARC^{TM} T1 and T2 employ chip multi-threading (CMT) technology to achieve high throughput on commercial workloads by taking advantage of the TLP inherent to such workloads.

OpenSPARC is an open source community based around hardware design and experimentation aids for UltraSPARC^{TM} T1 and T2. OpenSPARC provides open source availability of complete micro-architecture specifications, Verilog RTL code, a full suite of RTL and architectural simulations and infrastructure, FPGA implementations of the microprocessors, reference boards with microprocessors, hypervisor code and multiple operating system ports [1].

This paper briefly describes the UltraSPARC^{TM} T1 and T2 architecture upon which OpenSPARC is based. The reliability and error management features of the architecture are discussed, especially the benefits of lower temperature operation achieved through reduced power consumption. Some ongoing research projects in hardware reliability at universities designated as OpenSPARC Centers of Excellence – Carnegie Mellon University, University of Illinois at Urbana-Champaign and Stanford University – are described along with their experiences in using OpenSPARC for the research. Future research directions in hardware reliability, error management, fault tolerance and on-line test based on OpenSPARC are proposed in the final section.

II. OpenSPARC Architecture

OpenSPARC is based on Sun's UltraSPARC^{TM} T1 and T2 microprocessors. The UltraSPARC^{TM} T2 processor is the industry's first "server on a chip", with 8 cores, 64 threads, and on-chip networking and security functionality. The UltraSPARC^{TM} T1 processor is an earlier version with 8 cores and 32 threads and is described in [3],[4].

A block diagram of UltraSPARC^{TM} T2 is shown in Fig. 1. Each of the 8 SPARC processor cores has full hardware support for execution of eight independent threads. Each SPARC processor core consists of two integer execution units, a floating point and graphics unit, and a cryptographic stream processing unit. The eight SPARC processor cores share an 8-banked, 4MB Level 2 cache (L2). Each bank of the L2 cache is 16-way set associative with a line size of 64 bytes. The eight banks of L2 allow eight simultaneous accesses to support the high bandwidth requirements of the UltraSPARC^{TM} T2.

The SPARC processor cores communicate to the L2 cache through a high bandwidth, non-blocking, pipelined crossbar switch. The L2 cache connects to four on-chip DRAM controllers that interface directly to a pair of fully buffered (FBDIMM) channels. UltraSPARC^{TM} T2 features an on-chip PCI-Express unit and two on-chip 10 Gb/sec Ethernet ports with XAUI Interfaces, making it a true system-on-a-chip.

[*]The author is also affiliated with the School of Computer and Communication Sciences, École Polytechnique Fédérale de Lausanne.

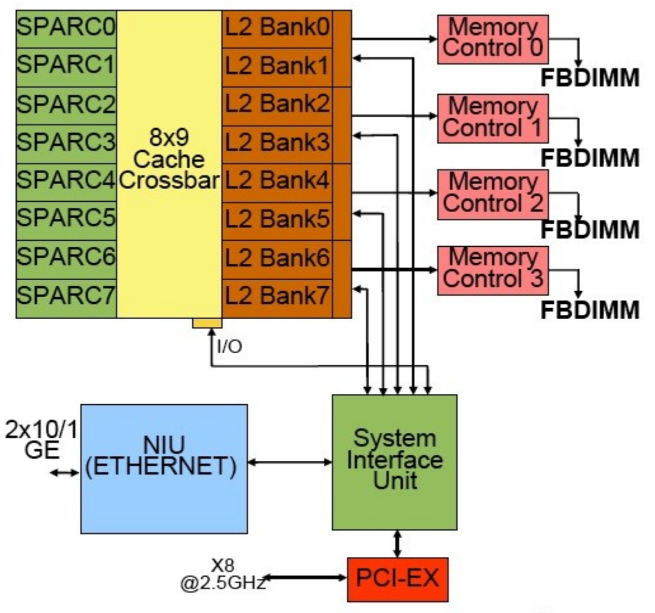

Fig. 1. UltraSPARC T2 processor. The diagram shows the high-level blocks and in the processor and their connectivity.

UltraSPARC^{TM} T2 implements very limited execution speculation in order to minimize overall power consumption. To further reduce power, UltraSPARC^{TM} T2 transitions a thread to a not-ready state whenever a thread encounters a long latency operation. Clock power is further reduced on UltraSPARC^{TM} T2 by utilizing clock gating on datapath, control, and array structures. Power throttling is supported through 3 external power throttle pins. Based on the state of the power throttle pins, stall cycles are injected into the processor core pipeline to reduce overall dynamic power consumption.

At the circuit level, the design was kept fully static to minimize power consumption. Extensive coarse and fine-grain clock gating techniques were used in the core, clusters and datapaths, where up to 30% of the flops are kept disabled at any time. In order to reduce static or leakage power, logic gates with sufficient timing, noise, and slew margins are replaced with footprint-compatible gates with longer channel length [6]. UltraSPARC^{TM} T2 includes on-chip thermal diodes, which enable the system to regulate the die temperature by controlling the instruction issue rate and by disabling threads.

III. Reliability in the OpenSPARC Processors

The OpenSPARC CMT architecture provides a major reliability benefit due to its reduced temperature gradients. Single-core processors usually have several hot spots at over 100ºC and with 30ºC–50ºC thermal gradients presenting cooling challenges and requiring special complex thermal management techniques [8]. In contrast, the distributed workload across multiple cores and the symmetrical floorplan allows a very uniform power density, resulting in not only a cool junction temperature but also in a low thermal gradient. Silicon measurements running typical workload applications show a worst-case junction temperature of 66ºC and a 7ºC thermal gradient. The hot spots in the die are located in between the two rows of cores, where the thermal diodes are placed to monitor and regulate power.

Overall chip reliability is enhanced since silicon reliability mechanisms, namely wearout and drift, are highly dependent on temperature. Wearout mechanisms, like gate-oxide integrity (GOI) and electromigration (EM), can cause devices to fail abruptly, leading to an immediate loss of functionality. As the worst case junction temperature reduces from typically 105ºC to a cooler 66ºC, the median GOI time-to-breakdown increases by a factor of nine while the average failure rate reduces by a factor of 17 [3]. Drift mechanisms, like negative bias temperature instability (NBTI) and channel hot carrier (CHC), degrade silicon devices over time, impacting performance, and causing shifted beta ratios, decreased headroom, mismatch, and increased delay variations [9]. As with wearout mechanisms, drift mechanisms also show a significant reliability improvement from lower operating temperature. For example, NBTI degradation improves by 29% at the lower junction temperatures, equivalent to an eight-fold increase in lifetime [4].

The OpenSPARC processors have active power control features implemented in both hardware and software that allow individual threads, or even cores, to be idled. Using this very fine-grained control of power and activity in combination with their inherent redundancy, multi-core CMT designs can act as reliability-aware processors [10] In a reliability-aware processor, active power control can be generalized to active temperature control. For instance, the processor could throttle power to limit high temperature excursions due to system or environmental issues. While limiting the worst-case temperature seen by a given processor obviously improves its reliability, this feature can also be used to optimize the performance-reliability trade-off of all processors.

This concept can be taken even further to allow individual processors to adapt to their own environments. For example, while most processors are located in well-controlled environments such as datacenters, some processors may be located in more extreme environments with higher ambient temperatures. With environmental monitoring, the reliability-aware processor could individually optimize its performance while remaining within target reliability specifications. Finally, a reliability-aware CMT processor has the ability to dynamically monitor and perform self-testing to assess its on- going health. In the event that a problem develops, the processor could adjust or otherwise re-configure itself to maintain functionality, while alerting the system console that service may be required.

The OpenSPARC architecture contains significant protection for on-chip memory structures. As a general guideline, all memory arrays greater than 8KB are required to be protected by Single Error Correction/Double Error Detection Error Correcting Codes (SEC/DED ECC), while arrays greater than 2KB are required to have parity protection. Table I shows the protection mechanisms for the UltraSPARC^{TM} T2 on-chip memories.

Several mechanisms are used to protect main memory. Chipkill correction technology is implemented to withstand multi-bit memory errors within a DRAM device, including a failure that causes incorrect data on all data bits. The T2 is able to correct any error contained within a single memory nibble (4 bits), while detecting all errors contained within any two nibbles. Data is written to DIMM with a checksum appended. If a memory error occurs, then the data is immediately recovered by recalculating it from the checksum information. This approach allows the system to correct not only single-bit memory errors via standard ECC, but also 2, 3, and 4-bit errors, and even a whole x4 DRAM chip failure.

TABLE I
ON-CHIP ERROR PROTECTION

HW Structure	*Protection*	*Recovery*
Instruction Cache Tag	Parity	Hardware
Instruction Cache Data	Parity	Hardware
Windowed Integer Register File	SEC/DED ECC	Software
Floating Point Register File	SEC/DED ECC	Software
Data Cache Tag	Parity	Hardware
Data Cache Data	Parity	Hardware
Store Buffer Data	SEC/DED ECC	Software
Store Buffer Tag	Parity	Software
L2 Cache Data	SEC/DED ECC	Hardware
L2 Cache Tag	Parity	Hardware
L2 Cache Coherence	SEC ECC	Software

IV. WHAT IS AVAILABLE IN OPENSPARC FOR EXPERIMENTATION

Under OpenSPARC, a comprehensive set of tools and aids are available, which make it a rich, practical platform for a range of research topics.

A. Chip Design and Verification

OpenSPARC T2 chip source code is intended for members of the hardware engineering community that are experienced in chip design and verification. The download for hardware design and verification engineers includes: 1) Verilog RTL for OpenSPARC T2 design, 2) Verification environment for OpenSPARC T2, 3) Diagnostics tests for OpenSPARC T2, 4) Scripts and Sun internal tools needed to simulate the design and to do synthesis of the design, and 5) Open source tools needed to simulate the design.

The systems requirements for running this environment are: SPARC CPU based system with Solaris 9 or Solaris 10 OS and a C/C++ Compiler. Commercial EDA tools required to do design and verification work are not provided by OpenSPARC and will need to be obtained independently are Synopsys VCS© for Verilog simulation and Synopsys Design Compiler© for synthesis. In addition, for physical design work, commercial tools from EDA vendors would be required. Most EDA vendors have versions of their tools freely available for university research.

B. Architecture and Performance Modeling

The download for architects and software engineers includes: 1) SAM - SPARC Architecture Model (including source code), 2) *Legion* - Fast instruction accurate simulator for software developers (including source code), 3) SAM/*Legion* enhancements to copy files to/from simulated disk, 4) SAS - Instruction accurate SPARC Architecture Simulator (including source code), 5) OBP - Open Boot PROM source code, 6) Hypervisor source code, 7) Solaris Images for simulation, and 8) RST Trace Tool - RST is a trace format for SPARC instruction-level traces.

SAM (SPARC Architectural Model) is a full system simulator that is able to boot hypervisor, OBP (Open Boot PROM) and Solaris and run applications. It loads SAS (SPARC Architecture Simulator) as the OpenSPARC T2 simulator, so any modifications made in SAS get automatically reflected in SAM. SAM is useful for software bring - for instance, to debug Hypervisor/OBP/Solaris on a modified CPU implementation. SAM is also useful for performance analysis, both to generate traces and to connect with a performance model to perform execution driven simulation. SAM loads device models as dynamically linked libraries, and is useful for device driver development, and device RTL verification.

Legion is a fast instruction accurate simulator that provides a rapid means of developing and testing software functionality in the absence of actual hardware. Legion provides the fastest simulation environment for developing and testing SPARC Software. Firmware and Software developers will be the primary users of Legion simulation environment for the OpenSPARC T2. The RST Tools package consists of the trace format definition, a trace reader/writer library, and a trace viewer program.

C. Tools for Tuning and Debug

There are several tools available in the OpenSPARC distribution that are useful for performance tuning and debug, including 1) ATS - Automatic Tuning and Troubleshooting System, 2) *Corestat,* 3) *Discover,* and 4) *Thread Analyzer.*

Automatic Tuning and Troubleshooting System (ATS) is a binary re-optimization and recompilation tool used for tuning and troubleshooting applications.

Corestat is an online monitoring of core utilization for servers designed using the UltraSPARC^{TM} T1 processors. Core utilization is reported for all the available cores by aggregating the instructions executed by all the threads in that core. Its a perl script which forks cpustat command at run time and then aggregates the instruction count to derive the core utilization. An UltraSPARC^{TM} T2 core has 2 integer pipelines where as T1 core has 1 integer pipeline. An integer pipeline can best execute one instruction/cycle and hence the maximum core utilization is directly proportional to the frequency of the processor. For UltraSPARC^{TM} T2, corestat also reports FPU utilization.

Discover is Sun Microsystems' Memory Error Discovery Tool that is used by software developers to detect programming errors related to the allocation and use of program memory at runtime.

The *Thread Analyzer* is a tool that analyzes the execution of a multi-threaded program and checks for a variety of multi-threaded programming errors such as data races and deadlocks.

D. Tools for Software Development

Sun Studio 12 software is the premier development environment for the Solaris operating system. Optimized C, C++ and Fortran compilers, combined with Netbeans-based, IDE and other performance tools, enable Sun Studio 12 software to deliver high performance for single and multithreaded application development on the latest Sun hardware platforms. Sun Studio 12 software is also available for the Linux OS.

Binary Improvement Tool (BIT) works directly with SPARC binaries to instrument, optimize, and analyze them for performance or code coverage.

Faban is the consolidation of Sun's benchmark development and management knowledge and experience. It is a facility for developing and running benchmarks. It has two major components, the *Faban* harness and the *Faban* driver framework.

Simple Performance Optimization Tool (SPOT) produces a report on the performance of an application. The spot report contains detailed information about various common conditions that impact performance.

A static source code analysis and code scanning tool that identifies incompatible APIs between Linux and Solaris platform and helps to simplify and speed up migration project estimation and engineering is also available.

V. Current University Research in Hardware Reliability

The availability of a wide range of design and analysis tools spanning chip design and architecture (as described in section 4) make OpenSPARC an excellent platform for research in silicon error management. It is also a state-of-the-art microprocessor that provides a real-life test case with sufficient complexity for demonstrating research concepts. Several Universities are actively using OpenSPARC for their research. A few universities have been designated as OpenSPARC Centers of Excellence based on their involvement with OpenSPARC. This section gives examples of OpenSPARC use in research in reliability and silicon error management at three OpenSPARC Centers of Excellence – Carnegie Mellon University, University of Illinois at Urbana-Champaign and Stanford University.

A. Carnegie Mellon University

Researchers at CMU have come up with a fingerprinting based approach for error detection in microprocessor cores [11]. Architectural fingerprints compress architectural state updates into a small hash for periodic comparison across redundant processor cores. The fingerprint is accumulated over a contiguous interval of instructions. Before retirement, the cores exchange their respective fingerprint values to detect single event upsets that have affected instructions in the interval. Because a single fingerprint can cover multiple instructions, fingerprinting offers significant bandwidth savings over directly comparing execution results. Architectural fingerprints were evaluated using statistical soft fault injection in the single-core OpenSPARC T1 RTL model. The Verilog model was simulated using Synopsys VCS with custom Verilog PLI modules added for soft fault injection. For each workload, the RTL model was first executed in a fault-free environment to establish the fault-free fingerprint values. The model was then run repeatedly with statistical fault injection. PLI injects faults into several top-level and representative units in the OpenSPARC T1 design and the entire SPARC processor core was also studied. Each unit was exercised with seven multithreaded validation test programs from the OpenSPARC package. User-level redundant multithreading, both within and across cores, was also demonstrated in the OpenSPARC RTL. In both instances, architectural fingerprints provided on-line error detection and isolation [13].

Another body of research with a scheme called FIRST (Fingerprints In Reliability and Self Test) targets detection of emerging wearout faults in silicon [12],[13]. Studies show that wearout faults have a gradual onset, manifesting initially as timing faults before eventually leading to hard breakdown. FIRST detects wearout faults as they begin to change timing, but before they affect normal operation. FIRST uses either architectural fingerprints or existing design-for-test hardware (scanout chains) alongwith infrequent periodic tests under reduced frequency guardbands to observe the marginal timing behavior that is an indication of wearout. FIRST is a low-overhead, efficient methodology for detecting emerging wearout faults before they affect normal operation. A Verilog PLI wearout model was was applied to several units from the OpenSPARC T1 design to demonstrate potential fault sites. The design was modified to add scan chains and fingerprinting logic. Fault injection and simulation was then performed to evaluate the effectiveness of the fingerprinting approach.

B. University of Illinois at Urbana-Champaign

One of the research projects at UIUC, SWAT (**<u>S</u>**oft**<u>W</u>**are **<u>A</u>**nomaly **<u>T</u>**reatment) is a complete framework for error detection, diagnosis, recovery, and repair/reconfiguration for a variety of hardware failure modes with a customizable tradeoff in reliability, performance, power and area. This framework spans hardware and software domains and combines low-cost high-level detection with potentially more expensive, low-level diagnosis. Some of the ideas that form the basis of this research are described in [14],[15],[16]. One key piece of this research project is developing and validating micro-architecture-level fault models for microprocessors to achieve fidelity of low-level fault modeling at the speed of higher-level simulation. A full system micro-architectural simulator at the highest level is being interfaced with gate level and timing simulators for low-level functional and timing simulation using the OpenSPARC design and Cadence NCVerilog. The two simulators communicate through the Verilog VPI.

The diagnosis and recovery piece of this research requires the firmware to be able to perform several tasks such as 1) control the transfer of checkpoints between faulty and non-faulty domains, 2) run instruction sequences on specific processor cores, 3) control and/or read which processor unit is used by specific instruction sequences, 4) communicate recovery information to the OS, etc. The firmware is being implemented on the hypervisor accompanying the OpenSPARC distribution and will be run on SunFire T2000 multi-core systems designed around the T1/T2 processors. This implementation will allow for experimentation to measure overheads due to the firmware for faulty and fault-free cores.

Another research project deals with errors induced by semiconductor process variation. The project has the following areas of focus: 1) modeling process variation and variation-induced errors in both logic and memory structures, 2) designing multi-core micro-architectures to detect and tolerate variation-induced errors, and 3) developing new micro-architecture techniques to mitigate variation-induced errors. The model of proces variation and resulting variation-induced timing errors is called VARIUS [17]. Some of the work forming the basis of this research involves pipeline design for variation [18], adaptive body bias applications to mitigate variation [19], workload scheduling and power management under variation [20], and core pairing for reliability [21]. OpenSPARC is the experimentation vehicle for extensions to this work. Physical implementations of the OpenSPARC design are used to validate the accuracy of the models, and to evaluate the micro-architectures required to detect, tolerate and mitigate variation-induced errors. The OpenSPARC design can be taken through a physical design flow using commercial synthesis and physical design tools, and the effects of process variation on critical paths can be studied very easily. The OpenSPARC RTL source code is being modified to implement the novel error-detection, tolerance and mitigation micro-architectures The modified RTL combined with the simulation test benches and fault injection routines are being used to validate the effectiveness of the solutions.

C.Stanford University

Researchers at Stanford University are investigating on-line test schemes that target early prediction of aging and infant mortality related failures as well as self-test based error diagnosis *after* a failure has occurred. A **C**oncurrent, **A**utonomous chip self-test using **S**tored **P**atterns (CASP) has been developed, in which a multi-core processor chip tests itself by scheduling one or more cores for some form of self-test without bringing the system down and without any downtime visible to the end user [22]. The basic idea of CASP is to store high quality test patterns in non-volatile memory such as FLASH or hard disks in a system, and provide architectural and system-level support for testing one or more cores in a multi-core processor, while the rest of the system continues to operate normally. CASP uses existing on-chip DFT and test compression features that are used for production test.

The CASP concept has been demonstrated successfully using the OpenSPARC T1 processor. To support CASP, a CASP test controller and an on-chip buffer to store a scan test pattern and its corresponding expected response and mask were implemented. Architectural support necessary for CASP 1) *before a test,* such as stalling and draining the pipeline, disabling communication with core under test, saving critical states and invalidating L1 data cache, and 2) *after a test*, such as restoring critical states, enabling communication and restarting the pipeline, was built into OpenSPARC T1 processor. This was done by either reusing existing support or by modifying it for the purposes of CASP. All features were incorporated by adding or modifying approximately 8,000 lines of Verilog code out of the hundreds of thousands of lines in the original design. Most of the modification did not require major changes to the normal operation of the chip, which simplified the verification task. The functionality of CASP was verified by arbitrarily selecting a core for self-test during regression verification runs, applying tests to the selected core while the regression continued, resuming normal operation of the core under test, and matching final results after the regression tests completed.

VI.Future OpenSPARC Research Possibilities

In addition to the specific examples of ongoing research described in the previous section, there is a vast range of research topics where OpenSPARC could potentially be used. This section briefly describes some of the topics in the area of hardware reliability and silicon error management that are of interest to the research community in academia as well as the industry.

1) Taking advantage of CMT nature of processors for error detection and recovery solutions. The trend of designing multiple cores and threads on a single die has opened up possibilities of using some of these resources for error detection and recovery. The classical solution of designing in redundant hardware for reliability can now be turned around to explore how multiple instances of hardware already present on a chip can be reused for making hardware reliable. Using cores as *spare* cores, using dedicated threads for duplicating execution, and using spare threads for error management functions, are examples of the rich research topics.

2) Understanding errors in the context of micro-architectural features specific to CMT microprocessors. The trend toward multi-core, multi-threaded processors, has given rise to new architectural innovations such as automatic thread scheduling, thread arbitration, scout threading, transactional memory in hardware, etc. OpenSPARC provides and excellent base for studying the behavior of errors in the presence of these CMT architectural features.

3) Validating the efficiency of any error management solution. One of the main categories of research in hardware reliability continues to be techniques to detect and recover from errors. Since OpenSPARC represents a state-of-the-art real microprocessor design, it is an excellent vehicle to demonstrate the effectiveness of any error management solution.

4) Studying the impact of reliability solutions on performance. One of the questions that often arises is the impact of the reliability solutions on area, power and performance. Area and power impacts are relatively easy to estimate, but performance degradation is much more difficult to estimate since it requires actual simulation of traces and some performance modeling. OpenSPARC comes with a suite of performance modeling tools, which makes it is very

easy to estimate the impact of error handling solutions on performance.

5) Software and firmware solutions for hardware reliability. Most current research in the area of hardware reliability, focuses on solutions in either the hardware domain or the software domain. One reason for this approach is that there is no easy experimental setup for demonstrating a solution that spans the entire stack. OpenSPARC with its access to RTL source code, hypervisor code and FPGA implementation makes it possible to create an experimental setup where all hardware and software aspects can be studied and solutions validated.

6) Evaluate the impact on local error detection and correction logic on chip level failure rates. The OpenSPARC design has built in several error detectors that cover various portions of the chip as described in Section 3. These error detectors can be selectively disabled. The impact of individual detectors on chip level failure rates, especially at application level, can be studied for a variety of fault models.

VII. Conclusions

OpenSPARC is an open source community based around UltraSPARC™ T1 and T2 CMT processors. In this paper we have described the rich experimental infrastructure available in the OpenSPARC platform - micro-architecture specifications, Verilog RTL code, suite of RTL and architectural simulations and test benches, FPGA implementations, reference boards, hypervisor source code and multiple operating system ports. This rich infrastructure and error management support inherent in the T2 architecture make OpenSPARC the most practical, real and versatile platform for research in reliable computing. Several universities are already using OpenSPARC in successful research projects and there is great potential for hardware reliability research by the academic community using OpenSPARC.

Acknowledgment

The authors acknowledge all the university students who contributed to and are currently involved in OpenSPARC based research described in this paper.

References

[1] "OpenSPARC: World's First Free 64-bit Microprocessors", http://www.opensparc.net.

[2] M. Horowitz and W. Dally, "How scaling will change processor architecture," IEEE International Solid-State Circuits Conf. (ISSCC) Dig. Tech Papers, Feb. 2004.

[3] A. S. Leon, et al., "A Power-Efficient High Throughput 32-Thread SPARC Processor," IEEE Journal of Solid-State Circuits, Vol. 42, No. 1, January 2007.

[4] A. S. Leon et al., "The UltraSPARC™ T1 processor: CMT reliability," Proc. IEEE Custom Integrated Circuits Conf., Sep. 2006, pp. 555–562.

[5] M. Shah, et al, "UltraSPARC™ T2: A Highly-Threaded, Power-Efficient, SPARC SOC", IEEE Asian. Solid-State Circuits Conf, Nov. 2007.

[6] U. G. Nawathe, et al.. "An 8-Core 64 Thread 64b Power-Efficient SPARC SoC," IEEE International Solid-State Circuits, Dig. Tech Papers, Feb. 2007, pp. 108-110.

[7] G. Grohoski, "Niagara-2: A Highly Threaded Server-on-a-Chip,"18th Hot Chips Symposium, Aug., 2006.

[8] R. McGowen et al., "Power and temperature control on a 90-nm Itanium family processor," IEEE J. Solid-State Circuits, vol. 41, no. 1, pp. 229–237, Jan. 2006.

[9] T. Takayanagi et al., "A dual-core 64-bit UltraSPARC™ microprocessor for dense server applications," IEEE J. Solid-State Circuits, vol. 40, no. 1, pp. 7–18, Jan. 2005.

[10] J. Srinivasan, et al., "The Case for Lifetime Reliability-Aware Microprocessors", International Symposium on Computer Architecture, June 2004, pp. 276-287.

[11] J.C. Smolens, et al., "Fingerprinting: Bounding soft-error detection latency and bandwidth", IEEE Micro, vol 24, no. 6, Nov-Dec 2004, pp 22-29.

[12] J.C. Smolens, et al., "Detecting Emerging Wearout Faults", Third IEEE Workshop on Silicon Errors in Logic – System Effects (SELSE 3), April 2007.

[13] J.C. Smolens, "Fingerprinting: hash-based error detection in microprocessors", Ph.D. Thesis, Carnegie Mellon University, Pittsburg, PA, 2007.

[14] J. Srinivasan, et al., "Lifetime Reliability: Towards and Architectural Solution", IEEE Micro 25(30, 2005.

[15] X. Li, et al., "SoftArch: and architectural level tool for modeling and analyzing soft errrors", Proc, International Conference on Dependable Systems and Networks, pp. 496-505, July 2005.

[16] M. Li, et al., "Understanding the Propagation of Hard Errors to Software and Implications for Resilient System Design", to appear in Architectural Support for Programming Langauages and Operating Systems, 2008.

[17] S. Sarangi, et al., "VARIUS: A Model of Parameter Variation and Resulting Timing Errors for Microarchitects", IEEE Trans. On Semiconductor Manufacturing, February 2008.

[18] A. Tiwari, et al., "ReCycle: Pipeline Adaptation to Tolerate Process Variation", International Symposium on Computer Architecture, June 2007.

[19] R. Teodorescu, et al., "Mitigating Parameter Variation with Dynamic Fine-Grain Body Biasing", International Symposium on Microarchitecture, December 2007.

[20] R. Teodorescu and J. Torrellas, "Variation-Aware Application Scheduling and Power Management for CMPs", International Symposium on Computer Architecture, June 2008.

[21] B. Greskamp and J. Torrellas, "Paceline: Improving Single-Thread Performance in Nanoscale CMPs through Core Overclocking", International Conference on Parallel Architectures and Compilation Techniques, September 2007.

[22] Y. Li, et al., "CASP: Concurrent Autonomous Chip Self-test Using Stored Patterns", Proc. of Design & Test in Europe, February 2008.