Detecting Emerging Wearout Faults

Jared C. Smolens, Brian T. Gold, James C. Hoe, Babak Falsafi, and Ken Mai
Computer Architecture Laboratory
Carnegie Mellon University
Pittsburgh, PA 15213
http://www.ece.cmu.edu/∼truss

***Abstract*— Aggressive CMOS scaling accelerates transistor and interconnect wearout, resulting in shorter and less predictable lifetimes for microprocessors. Studies show that wearout faults have a gradual onset, manifesting initially as timing faults before eventually leading to hard breakdown. Prior work suggests detecting wearout faults as they begin to affect normal operation, but these techniques require complex circuit and/or microarchitectural changes. Our proposal, FIRST (Fingerprints In Reliability and Self Test), uses existing design-for-test hardware (scanout chains) and infrequent periodic tests under reduced frequency guardbands to observe marginal behavior that is an indication of wearout. FIRST is a low-overhead, complexity-effective methodology for detecting emerging wearout faults before they affect normal operation. We discuss the operation of FIRST error detection, present a model for wearout fault simulation, and demonstrate FIRST's effectiveness on a portion of a commercial microprocessor design.**

I. Introduction

As CMOS feature sizes continue to shrink, transistor and interconnect reliability worsens [1]. While numerous physical phenomena will account for future device failures, the common system-level impact is shorter and less predictable lifetimes for microprocessors [2].

Unlike traditional manufacturing defect and single-event upset fault models (e.g., stuck-at faults and transient bit flip, respectively), studies suggest wearout-related faults will appear with gradual onset and will first affect timing. Designers conservatively add timing slack—known as a frequency guardband—to ensure that logic meets latch setup times. As devices transition more slowly over their lifetime, combinational logic paths will eventually fail to meet timing requirements and encroach on the design's frequency guardband [3]. Initially faults will appear intermittent, depending on specific operating conditions (e.g., voltage, temperature, circuit inputs, etc.), but eventually result in permanent defects. The number of potential critical paths in complex designs—due to rising static variations [4]—make it difficult to predict which paths will fail first due to wearout.

Recent work advocates detecting and recovering from wearout faults as they occur during normal operation. We refer to such techniques as *just-in-time fault detection*. Prior proposals [5,6] integrate carefully designed fault detection mechanisms into existing designs, but face significant challenges from the increasing number of (unpredictable) critical paths [5] or require complex integration with existing resource scheduling mechanisms [6]. Bower et al. [7] avoid the challenges of circuit-level test, but require extensive design changes to support instruction-level checking.

In this paper, we propose *early fault detection* with FIRST: Fingerprinting in Reliability and Self Test. In FIRST, we propose infrequent, periodic (e.g., once daily) testing where application and system software are suspended from the core, and special test software runs to stress the microarchitecture and detect the onset of wearout. FIRST reduces the processor's effective frequency guardband while running test programs. By reducing the frequency guardband, marginal critical paths are exposed well before timing faults affect normal operation on those paths. Lightweight signatures of microarchitectural state, called microarchitectural fingerprints [8], provide detection using existing test hardware, without requiring advance knowledge of which devices will fail.

Our paper is organized as follows. We provide background on wearout faults in Section II. Section III introduces the FIRST concept and implementation. Section IV presents a simulation model for wearout faults. We evaluate the simulation model and FIRST methodology in Section V. We present related work in Section VI and conclude.

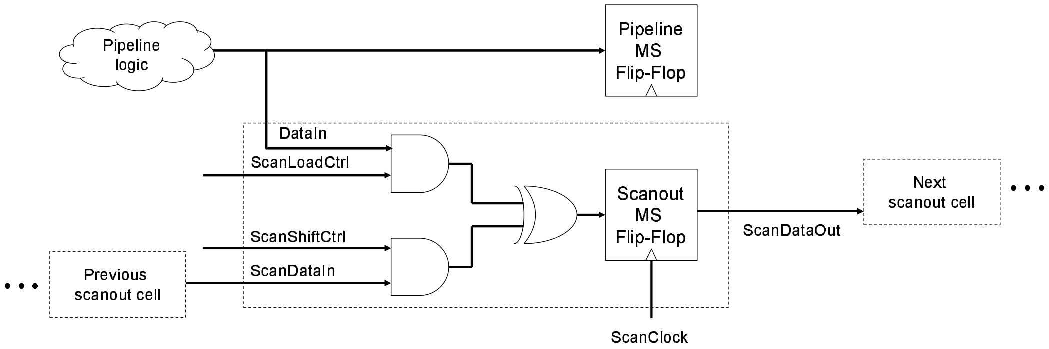
II. Background

Sources of wearout include gate oxide breakdown, hot-carrier injection, negative-bias temperature instability (NBTI), and electromigration [1]. The onset and end stages of device wearout (soft breakdown and hard breakdown, respectively) have been studied extensively in device reliability literature using accelerated wearout testing techniques.

During gate oxide soft breakdown, transistor switching speed decreases for a given operating voltage [3]. The logical operation of the transistor is nevertheless maintained [9]. This behavior means logic gate outputs transition more slowly as the soft breakdown progresses. Similar results have been shown for NBTI [10].

In light of this behavior, wearout appears as slow rising or falling transitions for certain inputs on the affected devices [11]. For example, a NAND gate with a failing NMOS transistor would experience a slowdown for the falling output, whereas a failing PMOS transistor would slow the rising output transition.

While today's designs have multiple statically known critical paths, increasing within-die variation associated with process scaling means that a particular die's critical paths are not necessarily known at design time [12]. Furthermore,



ScanShiftCtrl	ScanLoadCtrl	Operation
0	0	Reset
0	1	Snapshot DataIn
1	0	Shift chain
1	1	DataIn ⊕ ScanDataIn, shift chain

Fig. 1. A scanout cell records data or control-path signals on DataIn and hashes the monitored value with the output of previous cells in the chain.

experiments show that switching speed decreases dramatically during soft breakdown [10], which means that new critical paths may form. Therefore, predicting the location of wearout faults is difficult and detection mechanisms must look broadly across the design to detect the timing changes.

Although device wearout is gradual and precise failures are unpredictable, the failure rate has been shown to fit time-dependent distributions (e.g., log-normal and Weibull [3,13]). Furthermore, once a device has begun the failure process, research shows that the continued degradation rate is dependent upon operating conditions (e.g., exponentially related to supply voltage) [14]. With typical operating conditions, the soft breakdown occurs over days or weeks. Thus, there is opportunity for early detection of the first devices that begin soft breakdown.

III. FIRST

Based on the observation that common wearout faults exhibit a gradual onset, we propose the FIRST (Fingerprinting In Reliability and Self Test) methodology for early detection of wearout faults. The idea behind FIRST is to periodically test the processor core in near-marginal conditions to expose changes in timing that initially hide inside the processor's frequency guardband.

Effective early wearout detection demands (1) extensive coverage of circuit nodes to detect the first marginal devices and (2) mechanisms to induce marginal operation in the processor core.

The FIRST methodology performs in-field wearout fault detection in microprocessor cores. To start a test period, the operating system temporarily takes the core offline and places it in FIRST mode. The core then loads functional test programs that exercise logic transitions within control logic and datapaths. While the programs execute, the core generates a high-observability at-speed signature of internal microarchitectural state. The core repeats the same programs and signature collection as operating conditions are gradually changed (e.g., lowering supply voltage or increasing clock frequency), which effectively reduces the frequency guardband.

When the recorded signature no longer matches those of earlier executions, the frequency guardband has been exceeded and the test period ends. A signature mismatch at progressively more conservative operating conditions over subsequent test periods (e.g., over several days) indicates the onset of wearout. The appearance of a small number of errors presages extensive future failures and hard breakdown. The early warning allows time for scheduled replacement or removal of the failing processor.

A. Fault detection mechanism

Fault detection in FIRST is done by monitoring the cycle-by-cycle operation of the processor core using lightweight signatures of circuit node values. To generate the signatures in FIRST, we propose leveraging existing built-in processor test hardware for generating *microarchitectural fingerprints* of the processor's execution. A microarchitectural fingerprint is a small hash of internal state updates done by the processor over time (e.g., values stored in pipeline data and control registers). Fingerprints of a fixed functional test's execution can be compared across test periods and operational conditions to detect internal changes in the behavior of the processor that have not yet appeared in normal operation.

Microarchitectural fingerprints leverage existing manufacturing test and debug hardware called *scanout chains* to hash internal state over the course of execution [15]. Scanout chains are non-destructive sequential cells added to combinational and sequential nodes inside the processor core primarily to detect manufacturing defects and aid in at-speed silicon debugging. The logic cell illustrated in Figure 1 provides a *signature mode* that continually shifts state down the chain and XOR's it with the current value in the device. The output of the scanout chain can be fed into compactors such as linear feedback shift registers (LFSR) to create a summary of execution over space and time.

Because microarchitectural fingerprints monitor a large fraction of the internal state of the processor at full operating frequencies, they are useful for detecting changes due to timing and wearout faults. If the processor begins exhibiting mismatching fingerprints at progressively less stressed conditions (e.g., lower clock frequencies, higher voltages, and lower temperatures), this is an indication of the onset of wearout. The principal hardware for creating microarchitectural fingerprints is already available. Scanout chains provide a uniform test hardware across the design (as opposed to customized test hardware for each structure), and can be disabled when not in use (the scanout clock is independently controlled).

B. Inducing marginal operation

Several architectural and microarchitectural knobs are already available for artificially producing a near-marginal operating environment in the processor core, including dynamic clock frequency and width controls, voltage regulators with dynamic voltage scaling, and thermal monitoring and control [16].

Modern processors include clock scaling capabilities (to reduce power consumption during periods when the processor is idle). This capability can be employed to decrease the core's clock cycle time, and consequently reduce the guardband. Some processors also allow regional adjustment of clock skew and temporary phase shrinking and stretching [17].

Processors can also request changes in the voltage regulator output (also to save power). Lower voltages cause slower switching speeds that also serve to reduce the guardband for a particular clock frequency. Wearout faults have been empirically shown to increase frequency sensitivity to operating voltages [10].

On the monitoring side, processors contain thermal diodes to measure temperatures across the die. Coupled with functional test programs that run power-consuming instruction sequences, the diodes can establish a selected core temperature. Given the well-known relationship between switching speeds and temperature in CMOS, this environmental factor serves to temporarily reduce the guardband and expose wearout faults.

IV. Wearout Fault Modeling

A key challenge in evaluating microarchitecture-level wearout fault detection methodologies such as FIRST is accurately modeling the effects of wearout faults. In this section, we outline our wearout fault injection study and our simulation framework.

A. Fault Injection Study

Our overall goal is to understand the fault coverage characteristics of a range of in-field wearout fault mechanisms. Detection mechanisms range from simply checking the output of functional test programs to detailed observation of internal microarchitectural state. Our preliminary study focuses on the fault detection capabilities of microarchitectural fingerprints, which should provide high observability of emerging wearout faults.

Integral to the study of wearout faults is a model of wearout's effect on logic. In this paper, we present a preliminary simulation model of path delay faults. The model currently allows investigation of delay faults in a baseline circuit with pre-determined path delays. While this represents a static view of a circuit's path delays, we can control the clock period to model the effect of slowdown, or equivalently, the application of FIRST to a processor with a fixed degree of wearout. Our initial experiments with this infrastructure show the coverage of wearout fault detection using microarchitectural fingerprints. In future work, we will extend our framework to inject simulated wearout faults along paths within a processor. This model will allow evaluation of emerging wearout fault coverage.

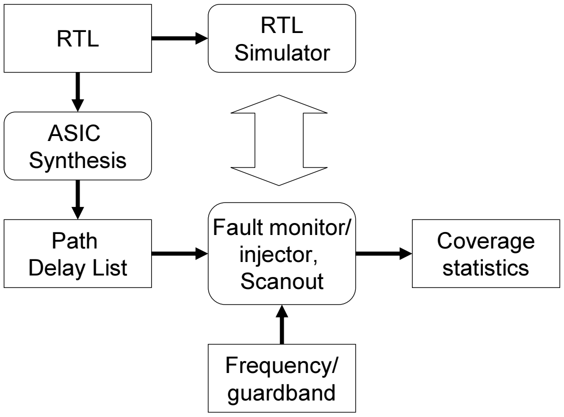


Fig. 2. The tool flow for modeling wearout faults.

B. Fault Simulation Background

Because wearout faults appear electrically as increased switching times, simulation models for wearout faults appear similar to path delay models used in manufacturing test to exercise critical paths [18]. In such tests, the circuit's critical paths are identified and logic input transitions are monitored for triggering conditions. On a matching transition during gate-level simulation, the affected logic output is forced to a stuck-at value to model the fault.

To maintain reasonable simulation speed, faults should be simulated in RTL (register-transfer level) instead of gate-level models. However, RTL generally models combinational and sequential logic without accounting for delay. Given wearout's similarities to path delay faults, delay fault simulation should provide an accurate model of wearout faults for coverage evaluation. We implement a technique similar to that used in SpeedGrade [19] to achieve the same accuracy as gate-level simulations for timing faults, but with simulation of RTL and timing fault trigger conditions.

C. Fault Simulator

We model wearout faults using the flow shown in Figure 2. The input is an RTL description of the circuit to be analyzed and a selected clock frequency (which determines the slack in the guardband), and the output consists of activation statistics for potential wearout fault sites and coverage of the detection mechanism.

From an RTL description, we first use ASIC synthesis and a standard cell library to generate a list of path delays in the circuit. A path is a sequence starting at a primary input (an output of a flip-flop), through a sequence of cells, to a primary output (the input of a flip-flop). The path delay list contains timing estimates for both rising and falling input transitions, based upon the standard cell library's characterization for each pair of primary inputs and outputs. This information is used to determine when a particular path is a candidate for missing timing in a baseline model of the circuit without wearout. For each circuit, the path delay list is static and only needs to be computed once.

Next, we simulate the circuit using an RTL simulator augmented with a custom wearout fault simulator and scanout chain model. The RTL simulator is a commercial Verilog simulator, while the fault simulator and scanout chains are C-language based libraries that communicate with the simulator through the Verilog programming language interface (PLI). The fault simulator works as follows. The path delay list is read for the circuit. The selected guardband, specified as a clock period in this model, is used to eliminate paths from consideration in later simulation. If it can be determined that the selected clock period is always long enough to meet timing for all possible input transitions on the path, the path can be eliminated from consideration (i.e., the transition delays with added wearout delays are less than the clock period).

On each simulated clock cycle, the fault simulator identifies transitions on primary inputs that *potentially* cause the primary output to miss timing. If timing is missed, the combinational logic driving the primary output should behave as if the primary input had never transitioned (i.e., stuck at the previous cycle's value). In this situation, the primary output determined by the RTL simulator may be incorrect with respect to the fault model. However, because the primary input may not be a controlling value, the primary output may still be unaffected by the delay fault.

To determine the correct values for primary outputs, the fault simulator temporarily undoes the input transition (by forcing the previous cycle's value on that node) and steps the Verilog model to update the corresponding primary outputs. The updated primary outputs are then compared with the original fault-free output to determine whether the fault was masked by other controlling values or, indeed, propagated to the primary output. The fault simulator then restores the original primary inputs. If the updated primary output does differ from the initial fault-free output, the fault simulator forces the primary output to its faulty value for the next cycle. The Verilog simulator then advances time and latches the primary outputs on the clock edge.

Finally, the fault simulator tracks statistics for each path delay, including whether any triggering conditions occurred for the path's primary input and whether those conditions were also propagated to the primary output.

V. Evaluation

In this section, we briefly characterize the wearout fault model and evaluate FIRST's detection capabilities using microarchitectural fingerprints. The objective of these preliminary experiments is to demonstrate that microarchitectural fingerprints provide high observability of small numbers of faults, as would be the case in the early stages of wearout.

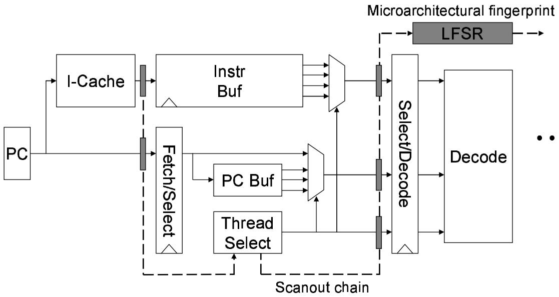


Fig. 3. Simplified front-end fetch, thread select, and decode stages of the OpenSPARC T1 pipeline with microarchitectural fingerprints added on data and control paths (shown in gray and with dashed lines).

We apply the wearout fault modeling tool flow from Section IV to the thread select unit of the Sun OpenSPARC T1 design [20]. The OpenSPARC T1 is a multi-threaded, multi-core chip design, made available in synthesizable Verilog RTL. The thread select logic controls the readiness state among four threads and selects one thread to execute on each cycle, based upon readiness, priority, and pipeline structural hazards. Figure 3 shows a simplified front-end of the OpenSPARC T1 pipeline. Scanout chains, in gray, are added to monitor important control and datapath signals in the design. The scanout chain terminates in an LFSR which contains the microarchitectural fingerprint of the execution.

We calculate the path delays using the Synopsys Design Compiler mapping to an Artisan/TSMC 0.18um low-power standard cell library. The longest transition delay in the unit is estimated to be 951ps over 6,929 paths between 186 flip-flop bits. We simulate the thread switch logic using Synopsys VCS. We model the scanout chains using Verilog PLI, which avoids changes to the original RTL and we accumulate the output in a 16-bit LFSR. We exercise the circuit with uniform random input vectors at the module level for 10,000 input vectors following a reset sequence (additional input vectors do not significantly change the results) and a cool-down time to shift remaining values out of the scanout chain. The modeled operating frequency is varied between 900ps and 955ps. We use the baseline circuit delays for all results.

The fault activation results are summarized in Table I. The

TABLE I
Fault activation results for the thread scheduler over a range of clock periods.

Clock Period (ps)	Possible Paths	Activated Paths	Propagated Paths
900	207	28	9
905	181	25	7
910	137	20	6
915	99	16	4
920	73	12	4
925	51	6	1
930	33	3	0
935	22	2	0
940	14	2	0
945	9	1	0
950	4	1	0
955	0	0	0

total number of potentially sensitized paths is listed in the second column. As discussed in Section IV, path delays which are always shorter than the clock period are discarded from consideration, therefore the number of sensitized paths increases with shorter clock periods. The third column indicates the paths which were activated. For most paths in this circuit, no transitions are observed at the primary inputs to activate the fault. Furthermore, because of logical masking even fewer of the activated paths are completely sensitized and propagate an incorrect value at the primary output (shown in the final column). The baseline circuit has 6,929 total paths, of which 2,459 and 1,029 can be activated and propagated, respectively. As shown in the table, a majority of these paths are shorter than 900ps. One important result of this experiment is that the statically determined longest paths do not necessarily determine the minimum usable clock period, even if activating transitions can be produced at primary inputs.

The microarchitectural fingerprints accumulated over each test period correctly distinguished executions where all timing has been met from those where a fault had been propagated to a primary output. That is, as the clock period is decreased, the fingerprint begins to mismatch the fault-free fingerprint at 925ps when the first fault propagates to a flip-flop. These results show that fingerprints differ even when only a single path has propagated an error during the test. The fingerprints clearly identify when one or a small number of paths in the design begin to miss timing. Therefore, microarchitectural fingerprints will provide the high fault observability needed for the FIRST methodology.

VI. Related Work in Wearout Detection

Recent research has produced a number of proposals for predicting, detecting and tolerating wearout faults in microprocessors.

Srinivasan et al. [21] applies known device failure models to the processor microarchitecture to monitor its mean-time-to-failure based upon dynamic operation. While this technique allows predictions of a processor's remaining operational lifetime, it does not detect actual wearout.

Dynamic error detection mechanisms actively look for errors in the field. A number of recent papers investigate just-in-time error detection, where errors resulting from wearout faults are detected during normal program operation. Bower et al. [7] utilize a DIVA [22] checker core to detect hard errors at runtime. The microarchitecture is extended with logic to track resources used by each instruction. Upon detection of an error by the checker, saturating counters in the units used by the instruction are incremented to indicate potentially failing units. This proposal has the advantage of also detecting and correcting soft errors with the DIVA checker. However, by utilizing existing test circuitry (scanout chains), FIRST requires far fewer changes to the microarchitecture.

The BulletProof pipeline integrates custom built-in self test (BIST) hardware with the processor pipeline [6]. When resources are idle during execution, test vectors are fed through functional units to detect faults that may have affected immediately prior execution. This integrates with a rollback-recovery mechanism to prevent potentially erroneous instructions from retiring. Unlike FIRST, the BulletProof pipeline requires extensive modification of the scheduling logic in the pipeline to run test vectors during idle resource cycles.

While proposed for aggressive voltage scaling, Razor can also potentially detect timing faults with double-sampled latches [5]. However, for effective wearout coverage, Razor would require special latches at each path where wearout may occur. The work of Blome et al. [23] proposes a circuit-level detection unit that monitors the delay of signals over time to detect the initial onset of soft breakdown. While this technique, like FIRST, is early wearout detection, it also requires a custom detection unit at each path where wearout may occur. Therefore, this technique can only detect evidence of wearout on a pre-selected set of paths in the circuit, which may miss the early effects wearout on unmonitored nodes.

VII. Conclusions

We proposed FIRST, an early detection mechanism for wearout faults. The FIRST methodology avoids the complex or area-inefficient detection mechanisms associated with just-in-time fault detection. We leverage the observation that wearout faults begin with timing degradation and worsen over time. FIRST makes use of existing design-for-test hardware—scanout chains with signature-mode capture—to observe wearout in slightly stressed operation. We presented a wearout fault simulation methodology and briefly evaluated the model on a portion of a commercial processor design.

Acknowledgements

This work is supported by NSF CAREER award CCF-0347568, NSF award ACI-0325802, Sloan and DoD/NDSEG fellowships, C2S2, MARCO, CyLab, and by grants and equipment from Intel Corporation.

References

[1] J. W. McPherson, "Reliability challenges for 45nm and beyond," in *Proceedings for the 43rd Annual Design Automation Conference (DAC)*, June 2006.

[2] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The impact of technology scaling on lifetime reliability," in *Proceedings of the International Conference on Dependable Systems and Networks (DSN)*, June 2004.

[3] Y.-H. Lee, N. Mielke, M. Agostinelli, S. Gupta, R. Lu, and W. McMahon, "Prediction of logic product failure due to thin-gate oxide breakdown," in *Proceedings of the 44th Annual International Reliability Physics Symposium*, June 2006.

[4] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, Nov-Dec 2005.

[5] D. Ernst, N. S. Kim, S. Das, S. Pant, rajeev Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 36)*, December 2003.

[6] S. Shyam, K. Constantinides, S. Phadke, V. Bertacco, and T. Austin, "Ultra low-cost protection for microprocessor pipelines," in *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct 2006.

[7] F. A. Bower, D. J. Sorin, and S. Ozev, "A mechanism for online diagnosis of hard faults in microprocessors," in *Proceedings of the 38th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 38)*, Dec 2005.

[8] J. C. Smolens, B. T. Gold, J. Kim, B. Falsafi, J. C. Hoe, and A. G. Nowatzyk, "Fingerprinting: Bounding soft-error detection latency and bandwidth," in *Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct 2004, pp. 224–234.

[9] A. Avellan and W. H. Krautschneider, "Impact of soft and hard breakdown on analog and digital circuits," *IEEE Transactions on Device and Materials Reliability*, vol. 4, no. 4, pp. 676–680, Dec 2004.

[10] V. Reddy, A. T. Krishnan, A. Marshall, J. Rodriguesz, S. Natarajan, T. Rost, and S. Krishnan, "Impact of negative bias temperature instability on digital circuit reliability," in *40th Annual International Reliability Physics Symposium*, 2002.

[11] J. R. Carter, S. Ozev, and D. J. Sorin, "Circuit-level modeling for concurrent testing of operational defects due to gate oxide breakdown," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05)*, 2005.

[12] S. Borkar, "Design challenges of technology scaling," *IEEE Micro*, vol. 19, no. 4, pp. 23–29, Jul-Aug 1999.

[13] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "Exploiting structural duplication for lifetime reliability enhancement," in *Proceedings of the 32nd International Symposium on Computer Architecture*, June 2005.

[14] B. P. Linder, J. H. Stathis, D. J. Frank, S. Lombardo, and A. Vayshenker, "Growth and scaling of oxide conduction after breakdown," in *Proceedings of the 41st Annual International Reliability Physics Symposium*, August 2003.

[15] R. Kuppuswamy, P. DesRosier, D. Feltham, R. Sheikh, and P. Thadikaran, "Full hold-scan systems in microprocessors: Cost/benefit analysis," *Intel Technology Journal*, vol. 8, no. 1, February 2004.

[16] S. Gochman, A. Mendelson, A. Haveh, and E. Rotem, "Introduction to intel core duo processor architecture," Intel, Tech. Rep., 2006.

[17] S. Rusu and S. Tam, "Clock generation and distribution for the first IA-64 microprocessor," in *Proceedings of the 47th International Solid State Circuits Conference*, 2000.

[18] S. Natarajan, S. Patil, and S. Chakravarty, "Path delay fault simulation on large industrial designs," in *Proceedings of the 24th IEEE VLSI Test Symposium*, August 2006.

[19] K. S. Kim, R. Jayabharathi, and C. Carstens, "SpeedGrade: an RTL path delay fault simulator," in *Proceedings of the 10th Annual Asian Test Symposium*, Nov 2001.

[20] *OpenSPARC T1 Microarchitecture Specification*, Sun Microsystems, Aug 2006.

[21] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The case for lifetime reliability-aware microprocessors," in *Proceedings of the 31st Annual International Symposium on Computer Architecture*, June 2004.

[22] T. M. Austin, "DIVA: A reliable substrate for deep submicron microarchitecture design," in *Proceedings of the 32nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 32)*, Nov. 1999.

[23] J. A. Blome, S. Feng, S. Gupta, and S. Mahlke, "Online timing analysis for wearout detection," in *2nd Workshop on Architectural Reliability (WAR-2)*, Dec 2006.