

Robust and Resilient Designs from the Bottom-Up: Technology, CAD, Circuit, and System Issues

Vijay Janapa Reddi

Electrical and Computer Engineering
The University of Texas at Austin
vj@ece.utexas.edu

David Z. Pan

Electrical and Computer Engineering
The University of Texas at Austin
dpan@ece.utexas.edu

Sani R. Nassif

Austin Research Lab
IBM Corporation
nassif@us.ibm.com

Keith A. Bowman

Circuit Research Lab
Intel Corporation
keith.a.bowman@intel.com

Abstract—The semiconductor industry is facing a critical research challenge: design future high performance and energy efficient systems while satisfying historical standards for reliability and lower costs. The primary cause of this challenge is device and circuit parameter variability, which results from the manufacturing process and system operation. As technology scales, the adverse impact of these variations on system-level metrics increases. In this paper, we describe an interdisciplinary effort toward robust and resilient designs that mitigate the effects of device and circuit parameter variations in order to enhance system performance, energy efficiency, and reliability. Collaboration between the technology, CAD, circuit, and system levels of the compute hierarchy can foster the development of cost-effective and efficient solutions.

I. INTRODUCTION

Variability in device and circuit parameters is emerging as a critical challenge in the semiconductor industry. Parameter variations degrade the performance, energy efficiency, and reliability of microprocessor products across all market segments, which range from the small embedded cores in a system-on-chip (SoC) to large multi-core servers. As technology scales, the effects of device and circuit parameter variability on system-level metrics is expected to worsen. For these reasons, error-tolerant designs will significantly enhance the value of future systems.

In this paper, we describe our collaborative vision for robust and resilient designs from the bottom process technology up to the system level of the hierarchy. In Section II, we highlight the technology challenges by summarizing the variability sources and projecting the failure probability for an SRAM bit cell in future generations. In order to reduce the manufacturing variability, we present recent advances in CAD tools in Section III. In collaboration with process technology and CAD, Section IV encompasses our review of the key trade-offs in error-detection and recovery circuits that can be made to enhance microprocessor performance and energy efficiency. Since a hardware-only solution is limited, we describe the potential system-level benefits of abstracting circuit-level information to the architecture and software layers in Section V. In Section VI, we conclude by emphasizing the potential opportunity for future robust and resilient system designs by exploiting the interaction of hardware and software.

II. TECHNOLOGY

The increase in manufacturing-induced variability causes the blurring of the boundary between catastrophic (hard) faults that are traditionally caused by topological changes in circuits due to manufacturing defects, and parametric (soft) faults related to device and interconnect variability. As this variability increases, it can cause circuits to exhibit faulty behavior similar to that caused by catastrophic defects. As an example, it has been the case for several technology generations that increasing threshold voltage variability in small MOSFET devices due to phenomena like Random Dopant Fluctuations (RDF) [1] and Line Edge Roughness (LER) [2] can cause a SRAM cell to be so asymmetric that it is only capable of storing one Boolean value instead of the two it can normally store. This causes the SRAM to behave as if it is *stuck* at one logic value, which will cause that particular SRAM cell to fail a write test. This type of phenomenon is already pervasive in SRAM at the 32nm node, and forward projections show that this type of behavior will continue, become more probable, and increase in scope to include other critical circuit types such as latches and register files. The sources of such failures may be categorized as follows:

- *Process Variations* — Statistical variations of device parameters such as channel length, threshold voltage, and carrier mobility. Channel length variability results from LER and other patterning fidelity issues, whereas threshold voltage variations are primarily induced by random dopant fluctuations.
- *Lifetime Variations* — Variations that shift physical parameters over the operating life of a circuit. The most significant are V_{th} shifts that are caused by negative-bias temperature instability (NBTI) and hot-carrier injection (HCI), and gate current shifts that are caused by time-dependent dielectric breakdown (TDDB).
- *External Noise* — Noise outside the external circuit, such as environmental/power supply noise and energetic particles that cause single-event-upsets.
- *Intrinsic Noise* — Noise sources inherent to normal device operation that become significant at small feature sizes. These include shot noise, thermal noise, and random telegraph noise.

To make predictions about the impact of these types of phenomena on future technologies, one of the authors (Nassif) has been working with International Technology Roadmap for Semiconductors (ITRS [3]) to develop an open predictive roadmap of circuit resilience metrics for future technologies [4]. The purpose behind the roadmap is to motivate research and development in this area by showing the potential future impact of scaling if no technology or circuit innovation takes place. To this end, the roadmap looks at three canonical *building block* parts of standard digital integrated circuits: an SRAM bitcell, a latch, and a CMOS inverter. These three circuits are good surrogates for major components of a digital CMOS design: storage (SRAM bitcell), synchronization (latch), and logic functions (inverter). Of course, it is common to have millions of SRAM bits, latches, and inverters in current high-performance microprocessors.

We obtained failure probabilities for the three canonical circuits in future high-performance (HP) technology nodes by simulating their behavior under the influence of manufacturing process variability, aging, and environmental fluctuations (primarily temperature and power supply). The simulations used the Predictive Technology Model (PTM) with variability estimates for both general logic and SRAM circuits for bulk CMOS technologies down to the 16nm node. We have been developing and refining this work constantly since the initial publication in [4], and here we will show some recent results specific to the SRAM.

We considered two distinct failure modes for the SRAM: (1) the writability fail, in which the SRAM is unable to store one of the two Boolean values, and (2) the read disturb fail, in which the act of reading an SRAM causes the contents of the cell to reverse polarity. In this roadmap study we attempt to predict the impact of scaling independent of other possible effects. Therefore, we are developing a circuit scaling methodology to port designs from one technology to another using the same general circuit topology while making appropriate adjustments to device sizes, P to N-channel transistor ratios, timing, and so on. For example, we do not change the circuit topology from the standard six-transistor cell because changing this cell topology has been reported to be a difficult choice for future technologies [5].

Fig. 1 shows the failure rates for the SRAM as predicted via this roadmap. The x-axis shows the relevant technology nodes and the y-axis shows the estimated SRAM bit-cell failure rate due to manufacturing-induced variability. The figure shows four curves. The ones labelled *Write* and *Disturb* show the failure rates for a conventionally scaled SRAM bit cell due to write and read-disturb mechanisms, and the curves labelled *Write(Big)* and *Disturb(Big)* show the failure rates for an SRAM bit cell, which is approximately 40 percent larger. We can make the following observations using the figure:

- 1) SRAM failure rates, which are already a significant problem that requires extensive design intervention (such as the introduction of redundancy and error correction), will continue to be a problem and will require even more circuit and architectural innovations to combat

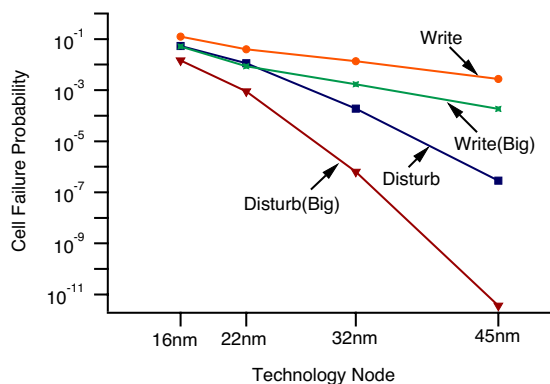


Fig. 1. SRAM variability-induced failure rates for various technologies.

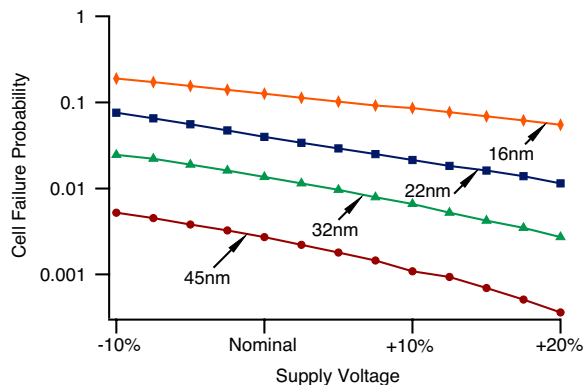


Fig. 2. SRAM write failures versus power supply voltage.

increasing manufacturing variability.

- 2) Enlarging the SRAM bit cell (reverse scaling) is moderately effective at controlling the impact of variability and may be used locally to create hardened portions of a design, but at a significant cost in layout density.

An important lever in reducing the impact of variability is the power supply. Designers would agree that raising the power supply voltage can significantly reduce circuit failure rates due to variability. Of course, this comes at the expense of additional power consumption, which is already a major factor for many types of designs. In order to understand the impact of power supply voltage on SRAM failure rates, we performed another study similar to that above in which we include the dependence of failure rates on power supply voltage. The range studied was from 10 percent below the nominal, which is a common worst-case assumption in digital circuit design, to 20 percent above the nominal. We show the results in Fig. 2, in which the x-axis is the power supply (as a percentage of the nominal supply for each technology node), and the y-axis is the probability of failure. To simplify the plot, we plot only the results for the SRAM Write failure. It is clear that the power supply has a *very* strong impact on SRAM Write failures and changes the failure rate by one order of magnitude over the range studies.

This observation leads to a clear engineering trade-off between power and robustness. Absent other sources of innovation at the device and circuit levels, one of the few effective levers for reducing circuit fail rates is power. But given that power itself is now one of the major design drivers, designers and technologists will have to take great care when developing balanced solutions for this problem.

III. COMPUTER AIDED DESIGN

Computer aided design tools, including modeling and optimizations, play very important roles in obtaining robust designs. For systematic variations that are often layout-dependent, researchers need to develop high-fidelity yet efficient models in order to capture those variations. These models can then be used to guide physical synthesis and mask synthesis to mitigate such variation-induced impact. Meanwhile, researchers shall develop effective variation-tolerant techniques for random variations in order to make the design tolerant to those unwanted variations. As explained in Section II, variations can result from the manufacturing process, aging, or operational conditions. In this section, we give examples in order to illustrate variational modeling issues and CAD optimizations.

A. Variational Modeling and Hotspot Detection

To support robust designs, we need to be able to model the variations, reliability, hotspots, and so forth efficiently, at desirable accuracy, and with the proper level of abstractions. To model the root causes of systematic lithography-induced variations, researchers proposed variational lithography modeling in [6]. This type of modeling was then used to perform true process variation aware optical proximity correction (PV-OPC). Variational lithography modeling takes into consideration two main sources of process variations: exposure dosage and focus. Based on the kernel decompositions, we can obtain closed-form models. To accurately model the electrical characterizations of non-ideal/non-rectangular gates, researchers often use slicing techniques [7], [8] to obtain the equivalent gate lengths.

As the feature size gets smaller, LER becomes more important because LER does not scale. LER depends on the development process, including exposure dose and thermal dose. Although LER is very much stochastic in nature, Ban et al [9] showed that it indeed displays certain layout-dependent variations. Thus, undesirable LER effects can be mitigated through certain layout optimizations.

For 22nm and 14nm nodes, double patterning lithography (DPL) becomes mandatory in order to use the mainstream 193nm immersion lithography to reach sub-80nm pitch size. Since DPL uses two masks to print dense features of the same layer, overlay is a major source of variations. For transistor layers, researchers observed the bimodal distribution [10]. For metal layers, overlay affects line width and spacing and therefore needs to be considered for parasitic extraction and timing analysis as well [11].

Due to extreme scaling, lithography hotspot (logic or electrical failure in a chip) is a key issue for physical verification. Existing works for lithography hotspot characterization/identification fall into two major categories: pattern matching techniques [12], [13] and machine learning/data mining [14]–[16] techniques. Although pattern matching methods yield high searching accuracy, they suffer from issues such as lack of pattern flexibility and explosion of pattern enumeration. On the other hand, machine learning based methods provide highly flexible models with low false alarms, but their accuracies still need to be improved. Most recently, [17] proposed a systematic meta-classification and detection that combines the strength of pattern matching and machine learning and showed very good trade-off.

B. Mask Synthesis

We can use variational modeling to guide more robust mask synthesis. Conventional mask synthesis, such as OPC, is not explicitly variation-aware or design aware. However, if we can pass the root-causes of these variations and their design impacts earlier on, mask synthesis can be more robust and cost effective.

For example, in [6], the compact variational lithography modeling is used to explicitly guide PV-OPC. It requires about 2-3 \times runtime of that for the conventional OPC but can reduce the CD error under variation significantly. This in turn reduces electrical performance (e.g., timing and leakage) variations. In [18], researchers proposed electrically driven OPC that minimizes the error in saturation current between printed and target shapes. Such design aware OPC can also help to reduce the mask complexity as not all the features need the same amount of extreme resolution enhancement.

One of the most fundamental problems with double patterning lithography is mask decomposition. To make the DPL tolerant to the overlay variations, [19] proposed a novel layout decomposition framework based on the *min-cut partitioning* formulation, in which balance density and overlay compensation can be modeled. Fig. 3 shows the effectiveness of the overlay compensation strategies during layout compensation. With a small penalty of stitch insertion, the overall timing variation of a critical path can be reduced significantly from over 9 percent to almost nothing. As the technology further scales to 14nm, 11nm, and beyond, new emerging lithography such as triple patterning lithography [20] and extreme ultraviolet lithography will require more robust mask synthesis to link the design and process more closely together.

C. Cell and Physical Designs

Standard cells are basic building blocks for VLSI designs. Thus, their robustness is essential. Typically, standard cell area is one of the most important metrics, along with other key electrical measurements like timing and power. On the other hand, area compaction, if not done properly, can make the cell more prone to variations. It is challenging to deal with area, performance, and DFM trade-off. In [21], researchers proposed a total-sensitivity based standard cell DFM flow. The

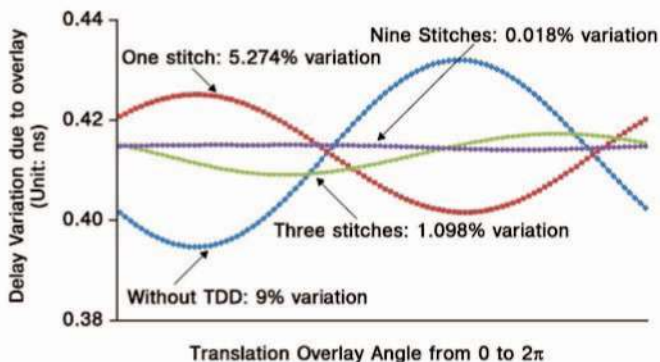


Fig. 3. DPL layout decomposition with overlay compensation.

sensitivity metric consists of the device/transistor criticality, the non-rectangular gate impact due to lithography printability, and process-variations (e.g., dosage and defocus). The total sensitivity modeling is built into a cell layout optimization engine to minimize the performance gap between process corners.

Routing is another key step that interacts with lower level manufacturing process variations. Many key manufacturing issues (e.g., topography variation due to chemical mechanical polishing or CMP, random defects, printability due to lithography limitations) are tightly coupled with interconnects/wires that are determined during routing. Holistic treatment must be used for manufacturability aware routing, including both the rule-based approach and the model-based approach. Different yield-loss mechanisms, such as topography variations after CMP [22], random defects [23], via failure [24], and lithography [25], [26] must be considered in a systematic manner. In [27], researchers suggest a possible DFM-aware routing flow (e.g., as shown in Fig. 4) in which different yield loss mechanisms are handled mainly at the stages that are more effective for them (e.g., global routing for CMP variation optimization, track routing for random defect minimization, construct-by-correction detailed routing for lithography hotspot minimization, post-routing optimization for redundant via insertion, and layer assignment/post-routing optimization for antenna effect mitigation). It may also be necessary to consider two or more manufacturability issues simultaneously (e.g., double patterning and redundant via together during routing [28], [29]).

Clock skew has global span in nature and, thus, is very sensitive to variations. Using the same principle, we should try to model and mitigate the predictable systematic variations as much as possible. For example, as shown in [30], the thermal profile can be used to make clock network more aware of the temperature distributions. Meanwhile, clock delay/skew will be susceptible to unbalanced aging. In [31], [32], researchers proposed novel circuit and CAD techniques for using NAND and NOR gates intelligently for clock gating such that the resulting NBTI-induced clock skew is minimized. To have ultimate robustness with respect to all sources of variations,

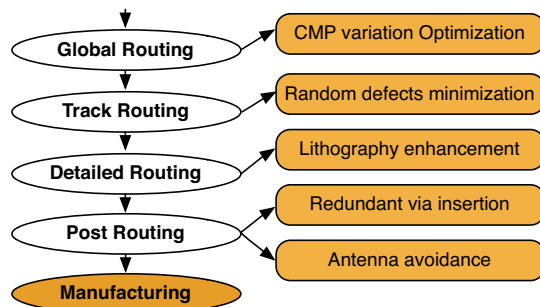


Fig. 4. A practical manufacturability aware routing flow.

cross link insertion [33], [34] and clock meshes [35] have to be used, which require careful planning and synthesis to obtain the skew target while minimizing the total wirelength/power.

IV. RESILIENT CIRCUIT DESIGN

Circuit designers work with process technology and CAD in order to gain a more accurate understanding of the variation sources and the impact of these variations on circuit performance, energy, and reliability. For each process technology node, advanced test-chip designs allow us to get direct measurements of process variations and lifetime variations, as described in Section II. From these measurements, we can construct variation models and apply them to CAD tools such as those discussed in Section III. These CAD tools provide the opportunity to explore a wide range of design trade-offs for optimizing circuit performance, energy, and reliability.

This section focuses on the design trade-offs for resilient error-detection and recovery circuits [36]. Conventional microprocessor designs apply clock frequency (F_{CLK}) and/or supply voltage (V_{CC}) guardbands to ensure correct functionality within the presence of worst-case dynamic variations. These inflexible designs cannot exploit opportunities for higher performance by increasing F_{CLK} or lower energy by reducing V_{CC} during favorable typical-case operating conditions. Since most systems usually operate at nominal conditions in which worst-case scenarios rarely occur, these infrequent dynamic parameter variations severely limit the performance and energy efficiency of conventional designs. Resilient circuits enable the microprocessor to operate at a higher F_{CLK} and/or lower V_{CC} as compared to a conventional design. When a dynamic parameter variation induces a timing error, the resilient circuits detect and correct the error. Although error recovery requires additional clock cycles, the gains from mitigating the guardbands for infrequent dynamic variations far outweigh the recovery overhead, resulting in higher overall performance and energy efficiency [36], [37].

A. Error Detection

In this subsection, we review two circuit designs for timing-error detection: (1) the embedded error-detection sequential (EDS) circuit design [36]–[43] and (2) the tunable replica circuit (TRC) design [36]. In the embedded EDS design, an EDS circuit replaces the receiving flip-flop of critical paths to detect

late timing transitions during the high clock phase, which defines the error-detection window. The fundamental trade-off in the EDS circuit is max-delay versus min-delay. The width of the error-detection window determines the maximum potential benefit in F_{CLK} as well as the min-delay penalty. For a target error-detection window, pre-silicon design satisfies the min-delay requirements with buffer insertion and sizing. In addition, a scan-configurable duty-cycle control circuit allows post-silicon tuning of the high clock phase to maximize the error-detection window while avoiding min-delay errors. An OR tree combines the error signals from each EDS circuit in a pipeline stage to produce a single pipeline-error signal, which is pipelined to the write-back stage to invalidate the errant instruction and to initiate error recovery [36].

In comparison to the embedded EDS design, the TRC design is less intrusive [36]. The TRC detects timing failures for a scan-configurable buffer delay chain with the input transitioning every cycle. A TRC is placed adjacent to each pipeline stage. Post-silicon calibration of the buffer delay chain ensures the TRC always fails if any critical path fails in the pipeline stage due to a dynamic variation. Thus, the TRC is tuned slower than the critical path. The TRC and the paths in the pipeline stage use the same local V_{CC} and clock, which enables the TRC to track critical-path delay changes due to V_{CC} droops while capturing clock-to-data correlations. If a dynamic variation induces a late timing transition in the TRC, the circuit generates an error signal that represents the single pipeline-error signal. As with the embedded EDS design, the single pipeline-error signal is pipelined to the write-back stage to invalidate the errant instruction and to enable recovery.

Table I lists the key trade-offs between the embedded EDS and TRC designs. The EDS design detects critical-path timing failures for fast, slow, long-range, and local dynamic variations. In contrast, the TRC design cannot detect path-specific or highly-localized dynamic variations (e.g., delay push-out from cross-coupling capacitance). The TRC requires a delay guardband to ensure the TRC delay is always slower than critical-path delays, thus preventing the possibility of exploiting path-activation rates for higher performance. Furthermore, the TRC design may initiate an error recovery when an actual error did not occur. This results in unnecessary recovery cycles. In comparison to the EDS design, the TRC design significantly reduces the design complexity overhead. In particular, the TRC design does not affect the min-delay paths in the core, has lower clocking energy, and does not require a duty-cycle control circuit. While the core min-delay constraints limit the error-detection window for the EDS design and, consequently, the maximum potential benefits, the TRC design provides a larger error-detection window to detect a wider range of dynamic delay variation. Both designs require post-silicon calibration, which affects testing costs.

B. Error Recovery

In this subsection, we review two techniques for error recovery: (1) instruction replay at $\frac{1}{2} F_{CLK}$ [36]–[38], and

TABLE I
EDS VERSUS TRC DESIGN TRADE-OFFS [36]

	EDS Design	TRC Design
Dynamic Variation Detected	Slow & Fast, Long-Range & Local	Slow & Fast, Long-Range
Exploit Path Activation	<u>Yes</u>	No
False Error Recovery	<u>No</u>	Yes
Design Complexity	High	<u>Low</u>
Min-Delay Overhead	Yes	<u>No</u>
Clock Energy Overhead	Yes	<u>Negligible</u>
Clock Duty-Cycle Control	Required	<u>Not Required</u>
Maximum Potential Benefit	Limited by Core Min-Delay Paths	<u>Not Limited by Core Min-Delay Paths</u>
Post-Si Calibration	Clock Duty-Cycle	TRC Delays

(2) multiple-issue instruction replay at F_{CLK} [36]. Reducing F_{CLK} in half ensures that the replayed instruction executes correctly even if dynamic variations persist. When initiating an error recovery, the error-control unit (ECU) signals the clock generator to reduce F_{CLK} in half via a clock-divider circuit to avoid relocking of the phase-locked loop (PLL). While F_{CLK} reduces in half, the duty-cycle control circuit maintains a constant high-phase clock delay to provide min-delay protection for the embedded EDS design. After the replayed instruction finishes, the ECU signals the clock generator to resume at the target F_{CLK} .

The motivation for the multiple-issue instruction replay design is to correct the errant instruction without changing F_{CLK} . This algorithm issues the errant instruction multiple (N) times. The first $N-1$ issues are replica instructions, which do not affect the architecture state. The N^{th} issue is a valid instruction, which is allowed to commit data to the architectural state. The replica instructions flow through the pipeline to set up the register nodes for the valid instruction. Any error that occurs in the execution of the replica instructions is ignored, and if the number of replica instructions is sufficient, the register inputs for each pipeline stage statically settle to the correct value. This allows the valid instruction to execute correctly. If an insufficient number of replica instructions are issued such that an error occurs during the execution of the valid instruction, then the errant instruction is replayed a 2^{nd} time with N equal to the number of pipeline stages to guarantee correct operation. Since this error-recovery design relies on setting up path nodes, this technique is directly applicable to static-CMOS logic circuits. This technique is not applicable to dynamic logic circuits.

C. Measurements

A 45nm microprocessor core integrates the resilient error-detection and recovery circuits presented in the subsections

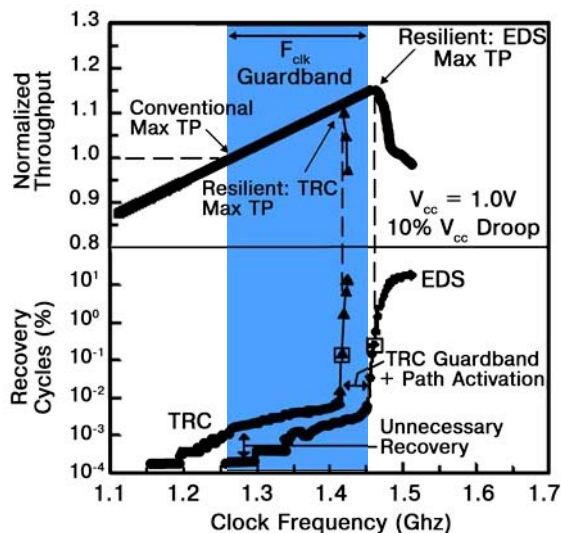


Fig. 5. Measured throughput normalized to the conventional maximum throughput, and recovery cycles as a percentage of total cycles versus F_{CLK} [36].

IV-A and IV-B to allow a direct comparison of the relative trade-offs [36]. In Fig. 5, the core without resilient circuits (i.e., conventional design) operates at a maximum clock frequency (F_{MAX}) of 1.45GHz at 1.0V. When a 10 percent V_{CC} droop occurs, the F_{MAX} reduces to 1.26GHz. As illustrated by the shaded region in Fig. 5, the difference between these two F_{MAX} values represents the F_{CLK} guardband for a 10 percent V_{CC} droop in the conventional design. Enabling EDS or TRC allows the infrequent errors from the V_{CC} droop to be detected and corrected, resulting in a higher F_{CLK} and throughput (TP). The optimal F_{CLK} for the resilient designs (1.46GHz for EDS, 1.42GHz for TRC) occurs at the point of maximum TP. Pushing F_{CLK} beyond this point reduces TP because the increasing number of recovery cycles outweighs the benefit of a larger F_{CLK} . In comparison to the conventional design, EDS and TRC circuits improve TP by 16 percent and 12 percent, respectively, at 1.0V. Although the EDS design provides a larger benefit at 1.0V, the error-detection window, and corresponding potential TP gain, is limited by core min-delay constraints. In contrast, the core min-delay constraints do not limit the error-detection window for the TRC design, which allows the TRC design to capture a wider range of dynamic delay variation. In Fig. 6 at low V_{CC} , the impact of variations increases and TRC provides more throughput gain than EDS (51 percent vs. 28 percent at 0.6V).

Fig. 7 evaluates the two error-recovery techniques by comparing the average number of recovery cycles per error. The figure shows measurements for the instruction replay at $\frac{1}{2} F_{CLK}$ and the multiple-issue (MI) instruction replay at F_{CLK} with N representing the number of issues. Measured performance for the core pipeline demonstrates that issuing only one replica instruction incurs the least number of recovery cycles per error and results in a 46 percent reduction as compared to replaying at $\frac{1}{2} F_{CLK}$. While the reduced number of recovery cycles per error improves performance, the salient

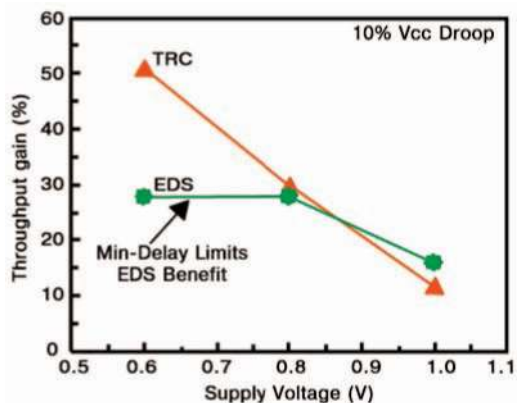


Fig. 6. Measured throughput gain versus V_{CC} [36].

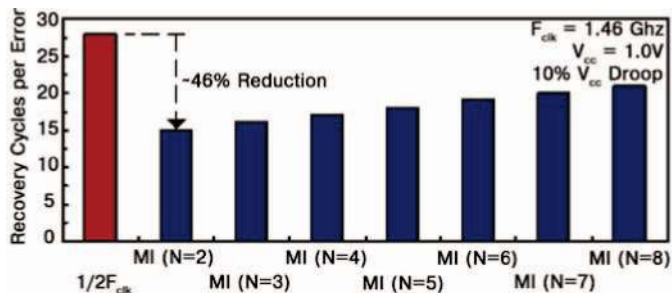


Fig. 7. Error recovery circuit analysis. Measured average recovery cycles per error for instruction replay at $\frac{1}{2} F_{CLK}$ (left-most bar) and multiple-issue instruction replay at F_{CLK} where N represents the number of issues [36].

advantage of the multiple-issue instruction replay is correcting errant instructions without changing F_{CLK} .

The resilient error-detection and recovery circuits mitigate the guardbands for infrequent dynamic variations to enhance performance and energy efficiency. Thus, the number of V_{CC} droops that occur during program execution for a specific V_{CC} droop magnitude directly affects the benefits of the resilient design. Fig. 8 plots the measured cumulative distribution of V_{CC} samples for 881 programs executing on the Intel[®] Core[™]2 Duo processor [44]. An oscilloscope probes the V_{CC} sense pins on the package in order to collect the V_{CC} noise measurements at a sampling frequency that is nearly equal to the core F_{CLK} (i.e., ~ 1 sample/cycle). The runtimes for the 881 programs range from 27 seconds to approximately 2 hours with an average runtime of around 30 minutes. The measurements in Fig. 8 provide insight into the V_{CC} droop occurrences during program execution. A detailed analysis indicates that the optimum recovery rate to maximize throughput in Fig. 5 corresponds to the V_{CC} droop sampling probability of 2.5×10^{-3} in Fig. 8. From these measurements, V_{CC} droop magnitudes that occur less than 0.25 percent ($V_{CC} \geq 3$ percent) are infrequent and, thus, can be effectively mitigated by the resilient circuits. However, V_{CC} droop magnitudes that occur greater than 0.25 percent ($V_{CC} \leq 3$ percent) are too frequent to reduce with the resilient hardware. In this case, the increase in the number of recovery cycles outweighs the F_{CLK} gain, resulting in lower throughput.

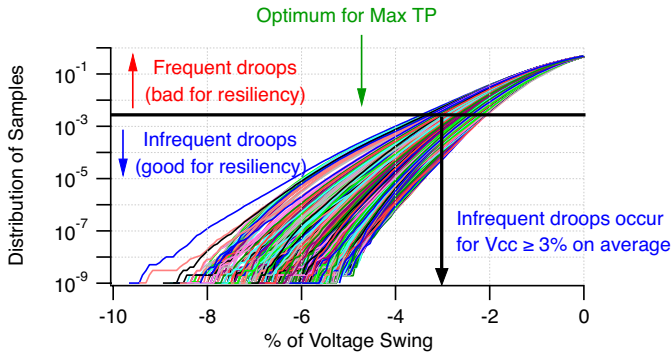


Fig. 8. Measured cumulative distribution of supply voltage (V_{CC}) samples versus the percentage of the nominal V_{CC} for 881 unique programs executing on the Intel[®] Core[™]2 Duo processor [44].

Frequent V_{CC} droops or persistent variations can result in long bursts of timing errors that degrade throughput. Thus, an adaptive clock control circuit can monitor the recovery cycles to optimize F_{CLK} for the current operating environment [36].

Although the resilient circuits ensure correct system operation within the presence of dynamic variations, a hardware-only solution is purely reactive. Allowing software, such as the operating system or some form of runtime layer, to monitor the recovery cycles in a resilient hardware might enable a more efficient system design by anticipating future events based on the workload. From Fig. 8, we can observe that the occurrence of V_{CC} droops varies widely across the different programs. By providing software the capability to monitor recovery cycles in the resilient hardware, we can track the optimum F_{CLK} setting for each workload, store these values, and then reuse this information during subsequent executions. In this way, software can predict the optimal F_{CLK} setting based on previous measurements to enhance the performance and energy efficiency of the system over the duration of its lifetime.

V. COLLABORATIVE SYSTEM DESIGN

In the future, we will require better integration and collaboration between hardware and software. As technology trends force us to build toward typical-case designs, error-detection and recovery mechanisms will become pervasive both in the microprocessor and system on chip designs. To sustain continued increases in performance, we must identify and develop new machine organizations that are capable of dynamically detecting and recovering from errors in the field across all layers of the computing stack, including computer architecture, system software, and applications. The benefits are two fold: (1) this eliminates performance inefficiencies that arise at each layer from maintaining strict abstraction between hardware and software, and (2) it eliminates power and area overheads that arise from the use of circuit- and microarchitectural-level techniques that mitigate the various sources of failures (see Section II).

We must look for ways to build reliable systems from unreliable devices using cross-layer solutions. Conventional

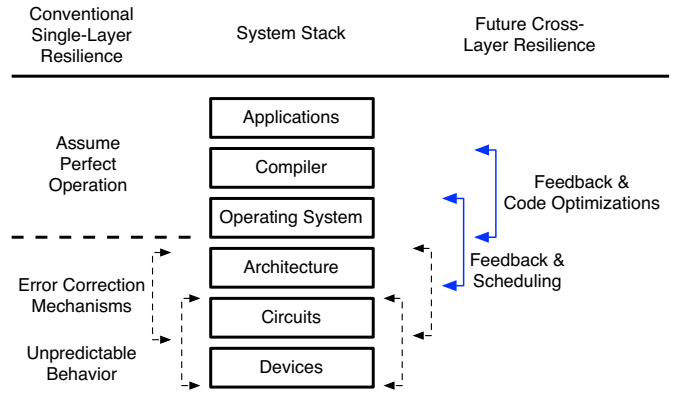


Fig. 9. Resilient system stack overview (adapted from [45], [46]).

belief is that hardware is perfect and software is “buggy.” However, we are beginning to accept the fact that this difference will blur towards software. A cross-layer solution will rely on hardware for fail-safe execution, but when it is possible it will attempt to optimize away recurring sources of errors via software techniques. Such an approach will eliminate recurring hardware penalties that arise from the hardware’s limited view of execution activity. For instance, in a large chip multiprocessor, one processor core might not have knowledge about another processor core’s activity even though each processor core interferes with the other’s work.

State-of-the-art resilient systems, such as that presented in Section IV, represent effort at a single layer. The left side of Fig. 9 (Conventional Single-Layer Resilience) illustrates the situation. Error detection and recovery at the hardware level guarantee correctness and hide any notion of failures and error recoveries from the software system stack. But therein lies the “Achilles’ heel” of the design: hardware-only solutions are *reactive* and are often difficult to implement efficiently. They introduce power, area, and performance overheads during typical-case operation for corner-case errors that occur very infrequently. In addition, they have limited knowledge about execution activity. Clock gating is one example of such an innovation. It is useful for power management, but this technique has led to new problems that require us to include additional circuits and microarchitectural techniques. For example, clock gating in high-performance processors can cause unacceptable stresses on the power delivery network. Although the resulting inductive noise, or $L \frac{di}{dt}$, can be handled using mechanisms to reduce voltage swings caused by large dynamic current, those mechanisms incur their own set of problems and performance impacts. The end result of this is a reactive approach to the design of high-performance, low-cost processors that resembles the children’s game “Whack-A-Mole”: see a problem, design a solution, optimize the solution, look for the next problem, and repeat.

The whack-a-mole situation is best understood by examining the $L \frac{di}{dt}$ effect closely. We explored voltage variation trends in the SPEC compute-intensive benchmark suite. We setup a framework similar to [47], in which we included a version of

the Watch architectural-level power simulator that calculated voltage variation every cycle [48]. The framework performed a convolution of Watch’s current estimates and an impulse response to the power-supply network for parameters derived from [49]. Resonance frequency for the power-supply network was 50 MHz. We simulated an 8-way superscalar, out-of-order processor. This setup allowed us to determine the number of voltage induced errors across different voltage margins.

We characterized the number of distinct errors that occurred during program execution, associating errors with static program code locations. Table II shows the number of different program counter locations and the absolute number of errors over a window of 100 million execution cycles. The total errors column represents the number of hardware error-recoveries that were required. The average number of total errors is large, in the thousands. Interestingly, however, the number of distinct program locations that map to these errors are significantly smaller, typically around tens of instructions. For example, one distinct program location in benchmark `ammp` is responsible for 94 error recoveries. In the case of the `swim` benchmark, we can avoid 218,188 error-recoveries if we remedy just 5 distinct program locations. This analysis *strongly* indicates that errors are tightly associated with few static program locations, which in turn is indicative of source code-level error-prone loops.

Ideally, if we could remedy all the static problematic program locations, then overall system reliability, performance, energy-efficiency would improve significantly because fewer hardware error-recoveries would be required. In this particular analysis, we simulated only 100 million instructions. But the case becomes even more compelling as we scale the number of instructions executed from 100 million to 1 and 10 billion. The number of distinct program locations stays roughly constant, while the number of errors increases with instruction count.

A. Software-Assisted Hardware Reliability

The goal is to mitigate reliability problems that can no longer be addressed efficiently at the device and circuit levels. Based on the data presented in the previous section, we claim that hardware-based solutions work well for intermittent errors that degrade reliability due to program activity. The best way to handle an error-prone program loop is through code transformations at the software layer. Typically, we have several options when creating the order of instructions in the software at the software layer, or when selecting the threads. Many of these options result in comparable system performance. The software layer seeks to eliminate emergencies from recurring in the future by giving hints to the hardware, code scheduling, and dynamic optimizations such as thread scheduling. Note that in such a system, the hardware-layer is responsible for guaranteeing reliable operation without the assistance of software. An advantage of this multi-layered approach is that it allows the hardware to focus on guaranteeing correct operation for the initial exceptional event while the software focuses on eliminating or reducing the performance impact in the future.

Currently, we do not comprehensively design software systems to receive error notifications from hardware. For instance,

when thermal emergencies activate hardware throttling, software can only query the processor for the number of cycles during which the processor was throttled. The software layer does not receive intelligent information about the instruction sequences or program architectural state. This is primarily because there is a general lack of understanding about machine activity and how that activity relates to code behavior that can trigger dangerous hardware events. Even if we were able to develop algorithms for identifying and optimizing potentially dangerous code behavior, such as that characterized by Gupta et al. [50] for voltage emergencies, the decision regarding whether or not to have software intervene would depend on the characteristics of the underlying system. For instance, we must know the characteristics of the power-supply network and operating voltage range of the target processor in order to cope with external sources of noise.

Therefore, we must extend the hardware so that it provides intelligent feedback to a runtime software layer that is dedicated to maintaining and managing system reliability. Such a dynamic optimizer must be inherently fine-grained, code-aware, and adaptive to the current execution environment. The software-based optimization framework can be implemented within a virtual machine monitor, a runtime compiler, or an operating system. By operating in *lazy optimization* mode the optimizer can wait until hardware intervention to determine whether a similar error has occurred in the past, and decide whether to make the active code region a candidate for error “optimization.” If required, then software can re-optimize the error-prone code and cache a version of the optimized code that exhibits more stable behavior. In an ideal case, only one iteration of an error will require hardware intervention (to guarantee fail-safe execution). All subsequent invocations of that code will execute the optimized error-free cache version.

The solid lines in Fig. 9 on the right side (Future Cross-Layer Resilience) provide a high-level view of a software-assisted hardware reliability management framework. The previously-proposed hardware-based voltage control mechanisms in Section IV remain intact, while new extensions are shown at the software level. The bottom layer includes low-level device and circuit blocks that provide sensed information to the architectural level. The architectural level collects all voltage sensor information that is filtered and combined with information about active code and thread sequences along with microarchitectural events before passing on the information to the software level. In the event of an error, the hardware control mechanism intercepts execution and performs various actions to correct the error. Simultaneously, the control mechanism provides feedback to the dynamic optimizer. Using feedback from the hardware, the optimizer performs the following five actions: (1) do root-cause analysis to identify the problematic program and error-prone code hotspot, (2) decide whether the identified error hotspot qualifies as a candidate for modifications, (3) determine the set of modifications to perform at the compiler or operating system level, (4) execute the best plan of action, and (5) resume execution.

Recently, researchers have done work that has demonstrated

TABLE II
NUMBER OF ERRORS VERSUS DISTINCT ERROR-PRONE PROGRAM LOCATIONS

SPEC Integer Benchmarks			SPEC Floating point Benchmarks		
Benchmark	Total Errors	Distinct Program Locations	Benchmark	Total Errors	Distinct Program Locations
gzip	57376	47	wupwise	54	4
vpr	45789	86	swim	218193	5
gcc	6346	64	art	59133	11
mcf	3525	37	mgrid	176668	24
crafty	201847	329	applu	119133	18
parser	28049	278	mesa	98509	102
eon	306698	40	galgel	58914	7
perlbmk	35	4	equake	119753	7
gap	6528	45	facerec	99140	14
vortex	139072	197	sixtrack	55234	88
bzip2	1284	12	ammp	94	1
twolf	98947	57	apsi	241056	37

much success in software-assisted hardware reliability. At the operating system level, researchers have demonstrated scheduling algorithms that lessen aging stress while taking process variations into account by intelligently mapping threads to multiple cores [51]–[55]. At the compiler-level, researchers have demonstrated how extrinsic power supply noise (or voltage variation) can be damped via instruction re-scheduling, both statically [56], [57] and dynamically [58]. Performing root-cause analysis is critical for several of these works, which is why anomaly detection has emerged as an important area of research [59]–[61]. These are all but a few samples that demonstrate that cross-layer software-assisted hardware reliability is feasible.

To facilitate cross-layer solutions at large-scale, we must first break the strict abstraction that the industry has driven between hardware and software. This effort will involve an interdisciplinary effort that spans from device physics to VLSI circuits to computer architecture and end-user system software and applications. The process of designing such a system will provide a new foundation for system design in which performance, reliability, and cost are all balanced goals.

VI. CONCLUSION

Reliability issues are real. While it may be feasible to stretch historically established industry techniques to combat reliability problems, such techniques are unlikely to be sustainable in the long-run. For example, we could continue to use F_{CLK} guardbands to address V_{CC} droops, but the degradation in power and performance will become larger in the future. Robust and resilient designs could mitigate such detrimental effects and provide significant benefits. Therefore, researchers should explore these new designs and also harness the synergy that arises from information exchange across the technology, CAD, circuit, and system layers. A synergistic approach provides us with a unique opportunity to optimize design trade-

offs from a system-level perspective. The benefits are that we can achieve new levels of cost-efficiency, and build efficient robust and resilient solutions that meet future demands for high performance, energy-efficiency and reliability.

REFERENCES

- [1] A. Asenov, “Random dopant induced threshold voltage lowering and fluctuations in sub-0.1 μm mosfet’s: A 3-d atomistic simulation study,” *IEEE Transactions on Electron Devices*, vol. 45, no. 12, 1998.
- [2] A. Asenov, S. Kaya, and A. R. Brown, “Intrinsic parameter fluctuations in decananometer mosfets introduced by gate line edge roughness,” *IEEE Transactions on Electron Devices*, vol. 50, no. 5, 2003.
- [3] [Online]. Available: <http://www.itrs.net>
- [4] S. R. Nassif, N. Mehta, and Y. Cao, “A resilience roadmap,” in *Proc. Design, Automation and Test in Eurpoe*, 2010.
- [5] L. Chang, D. Fried, J. Hergenrother, J. Sleight, R. Dennard, R. Montoye, L. Sekaric, S. McNab, A. Topol, C. Adams *et al.*, “Stable sram cell design for the 32 nm node and beyond,” in *IEEE Symp. on VLSI Technology*, 2005.
- [6] P. Yu, S. X. Shi, and D. Z. Pan, “Process variation aware OPC with variational lithography modeling,” in *Proc. Design Automation Conf.*, 2006.
- [7] W. J. Poppe, L. Capodiecici, J. Wu, and A. Neureuther, “From poly line to transistor: Building BSIM models for non-rectangular transistors,” in *Proc. of SPIE*, vol. 6156, 2006.
- [8] S. X. Shi, P. Yu, and D. Z. Pan, “A unified non-rectangular device and circuit simulation model for timing and power,” in *Proc. Int. Conf. on Computer Aided Design*, 2006.
- [9] Y. Ban and D. Z. Pan, “Modeling of layout aware line-edge roughness and poly optimization for leakage minimization,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, 2011.
- [10] K. Jeong and A. Kahng, “Timing analysis and optimization implications of bimodproc.cd distribution in double patterning lithography,” in *Proc. Asia and South Pacific Design Automation Conf.*, 2009.
- [11] J.-S. Yang and D. Z. Pan, “Overlay Aware Interconnect and Timing Variation Modeling for Double Patterning Technology,” in *Proc. Int. Conf. on Computer Aided Design*, 2008.
- [12] A. B. Kahng, C.-H. Park, and X. Xu, “Fast Dual Graph based Hotspot Detection,” in *Proc. of SPIE*, vol. 6349, 2006.
- [13] H. Yao, S. Sinha, C. C. Chiang, X. Hong, and Y. Cai, “Efficient Process Hotspot Detection using Range Pattern Matching,” in *Proc. Int. Conf. on Computer Aided Design*, 2006.
- [14] D. Ding, X. Wu, J. Ghosh, and D. Z. Pan, “Machine Learning based Lithographic Hotspot Detection with Critical Feature Extraction and Classification,” in *IEEE Int. Conf. on IC Design and Technology*, 2009.

- [15] D. G. Drmanac, F. Liu, and L.-C. Wang, "Predicting Variability in Nanoscale Lithography Processes," in *Proc. Design Automation Conf.*, 2009.
- [16] D. Ding, A. J. Torres, F. G. Pikus, and D. Z. Pan, "High performance lithographic hotspot detection using hierarchically refined machine learning," in *Proc. Asia and South Pacific Design Automation Conf.*, 2011.
- [17] D. Ding, B. Yu, J. Ghosh, and D. Z. Pan, "EPIC: Efficient prediction of ic manufacturing hotspots with a unified meta-classification formulation," in *Proc. Asia and South Pacific Design Automation Conf.*, 2012.
- [18] S. Banerjee, P. Elakkumanan, L. Liebmann, and M. Orshansky, "Electrically driven optical proximity correction based on linear programming," in *Proc. Int. Conf. on Computer Aided Design*, 2018.
- [19] J.-S. Yang, K. Lu, M. Cho, K. Yuan, and D. Z. Pan, "A new graph theoretic, multi-objective layout decomposition framework for double patterning lithography," in *Proc. Asia and South Pacific Design Automation Conf.*, 2010.
- [20] B. Yu, K. Yuan, B. Zhang, D. Ding, and D. Z. Pan, "Layout decomposition for triple patterning lithography," in *Proc. Int. Conf. on Computer Aided Design*, 2011.
- [21] Y. Ban, S. Sundareswaran, and D. Z. Pan, "Total sensitivity based DFM optimization of standard library cells," in *Proc. Int. Symp. on Physical Design*, 2010.
- [22] M. Cho, H. Xiang, R. Puri, and D. Z. Pan, "Wire density driven global routing for CMP variation and timing," in *Proc. Int. Conf. on Computer Aided Design*, 2006.
- [23] —, "TROY: Track Router with Yield-driven Wire Planning," in *Proc. Design Automation Conf.*, 2007.
- [24] G. Xu, L. Huang, D. Z. Pan, and M. D.-F. Wong, "Redundant-via enhanced maze routing for yield improvement," in *Proc. Asia and South Pacific Design Automation Conf.*, 2005.
- [25] J. Mitra, P. Yu, and D. Z. Pan, "RADAR: RET-aware detailed routing using fast lithography simulations," in *Proc. Design Automation Conf.*, 2005.
- [26] M. Cho, K. Yuan, Y. Ban, and D. Z. Pan, "ELIAD: Efficient Lithography Aware Detailed Router with Compact Printability Prediction," in *Proc. Design Automation Conf.*, 2008.
- [27] D. Z. Pan, M. Cho, and K. Yuan, "Manufacturability Aware Routing in Nanometer VLSI," in *Foundations and Trends in Electronic Design Automation*, 2010.
- [28] M. Cho, Y.-C. Ban, and D. Pan, "Double Patterning Technology Friendly Detailed Routing," in *Proc. Int. Conf. on Computer Aided Design*, 2008.
- [29] K. Yuan, K. Lu, and D. Z. Pan, "Double Patterning Lithography Friendly Detailed Routing with Redundant Via Consideration," in *Proc. Design Automation Conf.*, 2009.
- [30] M. Cho, S. Ahmed, and D. Z. Pan, "TACO: Temperature aware clock optimization," in *Proc. Int. Conf. on Computer Aided Design*, 2005.
- [31] A. Chakraborty, G. Ganesan, A. Rajaram, and D. Z. Pan, "Analysis and optimization of NBTI induced clock skew in gated clock trees," in *Proc. Design, Automation and Test in Eurpoe*, 2009.
- [32] A. Chakraborty and D. Z. Pan, "Skew management of NBTI impacted gated clock trees," in *Proc. Int. Symp. on Physical Design*, 2010.
- [33] A. Rajaram, J. Hu, and R. Mahapatra, "Reducing clock skew variability via cross links," in *Proc. Design Automation Conf.*, 2004.
- [34] A. Rajaram and D. Z. Pan, "Variation tolerant buffered clock network synthesis with cross links," in *Proc. Int. Symp. on Physical Design*, 2006.
- [35] —, "MeshWorks: An efficient framework for planning, synthesis and optimization of clock mesh networks," in *Proc. Asia and South Pacific Design Automation Conf.*, 2008.
- [36] K. Bowman, J. Tschanz, S. Lu, P. Aseron, M. Khellah, A. Raychowdhury, B. Geuskens, C. Tokunaga, C. Wilkerson, T. Karnik, and V. De, "A 45 nm resilient microprocessor core for dynamic variation tolerance," *IEEE Journal of Solid-State Circuits*, 2011.
- [37] K. Bowman, J. Tschanz, N. S. Kim, J. Lee, C. Wilkerson, S.-L. Lu, T. Karnik, and V. De, "Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance," *IEEE Journal of Solid-State Circuits*, 2009.
- [38] S. Das, C. Tokunaga, S. Pant, W.-H. Ma, S. Kalaiselvan, K. Lai, D. Bull, and D. Blaauw, "Razor II: In Situ Error Detection and Correction for PVT and SER Tolerance," *IEEE Journal of Solid-State Circuits*, 2009.
- [39] S. Das, D. Roberts, S. Lee, S. Pant, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "A self-tuning dvs processor using delay-error detection and correction," *IEEE Journal of Solid-State Circuits*, 2006.
- [40] P. Franco and E. J. McCluskey, "Delay testing of digital circuits by output waveform analysis," in *Proc. IEEE Int. Test Conf. on Test*, 1991.
- [41] P. Franco and E. McCluskey, "On-line delay testing of digital circuits," in *VLSI Test Symposium, 1994. Proceedings., 12th IEEE*, 1994.
- [42] M. Nicolaidis, "Time redundancy based soft-error tolerance to rescue nanometer technologies," in *VLSI Test Symposium, 1999. Proceedings. 17th IEEE*, 1999.
- [43] D. Ernst, N. S. Kim, S. Das, S. Pant, R. 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 *Proc. Annual IEEE/ACM Symp. on Microarchitecture*, 2003.
- [44] V. J. Reddi, S. Kanev, S. Campanoni, M. D. Smith, G.-Y. Wei, and D. Brooks, "Voltage smoothing: Characterizing and mitigating voltage noise in production processors using software-guided thread scheduling," in *Proc. Annual IEEE/ACM Int. Symp. on Microarchitecture*, 2010.
- [45] N. Carter, H. Naeimi, and D. Gardner, "Design techniques for cross-layer resilience," in *Proc. Design, Automation and Test in Eurpoe*, 2010.
- [46] V. J. Reddi, "Software-assisted hardware reliability: Enabling aggressive timing speculation using run-time feedback from hardware and software," Ph.D. dissertation, Harvard University, 2010.
- [47] R. Joseph, D. Brooks, and M. Martonosi, "Control techniques to eliminate voltage emergencies in high performance processors," in *Proc. Int. Symp. on High-Performance Computer Architecture*, 2003.
- [48] M. S. Gupta, J. L. Oatley, R. Joseph, G.-Y. Wei, and D. M. Brooks, "Understanding voltage variations in chip multiprocessors using a distributed power-delivery network," in *Proc. Design, Automation and Test in Eurpoe*, 2007.
- [49] S. Balasubramanian, "Power delivery for high performance microprocessors," in *Proc. Int. Symp. on Low Power Electronics and Design*, 2008.
- [50] M. S. Gupta, K. K. Rangan, M. D. Smith, G.-Y. Wei, and D. Brooks, "Towards a software approach to mitigate voltage emergencies," in *Proc. Int. Symp. on Low Power Electronics and Design*, 2007.
- [51] Y. Ding, M. Kandemir, M. J. Irwin, and P. Raghavan, "Adapting application mapping to systematic within-die process variations on chip multiprocessors," in *Proc. Int. Conf. on High Performance Embedded Architectures and Compilers*, 2009.
- [52] R. Teodorescu and J. Torrellas, "Variation-aware application scheduling and power management for chip multiprocessors," in *Proc. Annual Int. Symp. on Computer Architecture*, 2008.
- [53] A. Tiwari and J. Torrellas, "Facelift: Hiding and slowing down aging in multicores," in *Proc. Annual IEEE/ACM Int. Symp. on Microarchitecture*, 2008.
- [54] U. R. Karpuzcu, B. Greskamp, and J. Torrellas, "The bubblewrap many-core: popping cores for sequential acceleration," in *Proc. Annual IEEE/ACM Int. Symp. on Microarchitecture*, 2009.
- [55] Y. Li, O. Mutlu, and S. Mitra, "Operating system scheduling for efficient online self-test in robust systems," in *Proc. Int. Conf. on Computer-Aided Design*, 2009.
- [56] H. Tomiyama, T. Ishihara, A. Inoue, and H. Yasuura, "Instruction scheduling for power reduction in processor-based system design," in *Proc. Design, Automation and Test in Eurpoe*, 1998.
- [57] H.-S. Yun and J. Kim, "Power-aware modulo scheduling for high-performance vliw processors," in *Proc. Int. Symp. on Low Power Electronics and Design*, 2001.
- [58] V. J. Reddi, M. S. Gupta, M. D. Smith, G. yeon Wei, and D. Brooks, "Software assisted hardware reliability: Abstracting circuit-level challenges to the software stack," in *Proc. Design Automation Conf.*, 2009.
- [59] M. S. Gupta, K. K. Rangan, M. D. Smith, G.-Y. Wei, and D. Brooks, "Towards a software approach to mitigate voltage emergencies," in *Proc. Int. Symp. on Low Power Electronics and Design*, 2007.
- [60] M.-L. Li, P. Ramachandran, S. K. Sahoo, S. V. Adve, V. S. Adve, and Y. Zhou, "Understanding the propagation of hard errors to software and implications for resilient system design," in *Proc. Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, 2008.
- [61] S. Hari, M.-L. Li, P. Ramachandran, B. Choi, and S. Adve, "mSWAT: Low-cost hardware fault detection and diagnosis for multicore systems," in *Proc. Annual IEEE/ACM Int. Symp. on Microarchitecture*, 2009.