An Online Wear State Monitoring Methodology for Off-the-Shelf Embedded Processors

Srinath Arunachalam
Utah State University
srinath.arunachalam@aggiemail.usu.edu

Thidapat Chantem
Utah State University
tam.chantem@usu.edu

Robert P. Dick
University of Michigan, Ann Arbor
dickrp@umich.edu

Xiaobo Sharon Hu
University of Notre Dame
shu@nd.edu

ABSTRACT

The continued scaling of transistors has led to an exponential increase in on-chip power density, which has resulted in increasing temperature. In turn, the increase in temperature directly leads to the increase in the rate of wear of a processor. Negative-bias temperature instability (NBTI) is one of the most dominant integrated circuit (IC) failure mechanisms [13, 5] that strongly depends on temperature. NBTI manifests in the form of increased circuit delays which can lead to timing failures and processor crashes. The ability to monitor the wear progression of a processor due to NBTI is valuable when designing real-time embedded systems. While NBTI can be detected using wear state sensors, not all chips are equipped with these sensors because detecting wear due to NBTI requires modifications to the chip design and incurs area and power overhead. NBTI sensor data may also not be exposed to users in software. In addition, wear sensors cannot take into account variations in wear due to the differences in the wear sensor devices and the other functional devices and their operating conditions. In this paper, we propose a lightweight, online methodology to monitor the wear process due to NBTI for off-the-shelf embedded processors. Our proposed method requires neither data on the threshold voltage and critical paths nor additional hardware. Our methodology can also be extended to predict the wear progression due to some other dominant IC failure mechanisms. Experiments on embedded processors provide insights on NBTI wear progression over time. This knowledge can be used to design real-time embedded systems that explicitly consider runtime wear progression to increase predictability and maintain lifetime reliability requirements.

1. INTRODUCTION & CONTRIBUTIONS

Increasing device density resulting from process scaling has increased IC power density. The power wall has changed the way ICs are designed, as transistors have become inexpensive while power expensive. In addition to being power hungry, modern ICs have high power density and hence temperature. High temperature, along with process variations, exacerbates reliability, as microprocessor failure rate depends exponentially on operating temperature [19]. Temperature also affects speed; reduction of charge carrier mobility in transistors and increased interconnect latency resulting from high temperature degrade performance, potentially resulting in run-time failures. Worse, cooling technology has not improved at the same rate as transistor technology.

Embedded systems are used in a wide range of applications such as health care monitoring, aviation, and automobiles. Many embedded systems are required to be reliable; processors running embedded applications are expected to meet performance requirements for a certain amount of time even when operating under harsh conditions. In fact, an unexpected failure can lead to a catastrophe. Knowledge of the wear progression of embedded processors can greatly help in the design of reliable real-time embedded systems since components that are close to failure can readily be replaced and/or the activation of backup components planned.

There are several dominant IC failure mechanisms: electromigration (EM), stress migration (SM), time-dependent dielectric breakdown (TDDB), thermal cycling (TC), and NBTI. Of these, NBTI is one of the most dominant IC failure mechanisms [13, 5]. NBTI occurs when a negative bias is applied to a PMOS transistor, where an increase in temperature causes the threshold voltage to increase. The increase in threshold voltage increases the delay of the transistor, thereby, increasing the overall delay of the circuit. The increase in delay can cause timing violations, as well as permanent damage which can cause the processor to fail.

Wear progression due to NBTI can be observed since circuit delays worsen over time. NBTI manifests as timing faults in a processor. While several researchers have proposed using wear sensors [12, 11] to monitor and predict processor wear state due to NBTI, there are several drawbacks to this approach. First, many embedded processors do not have wear sensors. Second, wear sensors cannot account for the variations in wear due to the differences in the wear sensor devices and the other functional devices and their operating conditions. Third, wear sensor data may not be readily accessible to users and system designers.

In this article, we present a lightweight, online methodology that can be used to monitor the wear progression of a processor due to NBTI (and that of other failure mechanisms once difficult-to-find, model-specific parameters are known). The main idea behind our approach is to perform crash tests at specific voltage-frequency-temperature (VFT) points to detect timing violations that occur due to increased circuit delays. The crash tests are performed periodically to detect wear progression over time and are fully automated. Our proposed method targets off-the-shelf embedded processors

and works without wear state sensors or knowledge of the threshold voltage and critical paths. More importantly, wear progression over time can be observed using our methodology without requiring knowledge of the initial wear state of the system. Experimental results show a clear pattern of wear progression due to NBTI and that different processors have different initial wear states and age at different rates.

This paper is organized as follows. In Section 2, we review key existing research on online wear state monitoring and prediction. We discuss the reliability model in Section 3 and present our methodology in Section 4. Experimental results are provided and analyzed in Section 5. Section 6 concludes the paper.

2. RELATED WORK

Several techniques have been proposed to monitor wear due to NBTI in recent years [18, 12, 11]. The majority of these techniques require the addition of wear sensors either as separate circuitry or as part of the processor to monitor wear due to NBTI. For instance, an on-chip reliability monitor was proposed to measure NBTI degradation by measuring the beat frequency of two ring oscillators [12]. A phase comparator can then measure the difference in frequency between a stress and a reference oscillator to predict NBTI degradation. Another on-chip sensor uses a delay-locked loop for sensing on-chip NBTI degradation [11].

Researchers have also designed compensation circuits to negate the effects of NBTI. These circuits are highly specialized and require precise control during measurements. An on-chip slew rate monitor circuit was proposed to determine the degradation in the PMOS threshold by sensing the change in the rise time in a stressed ring oscillator [7]. By using slew rate instead of frequency, the impacts of NBTI can also be better quantified. An on-the-fly measurement technique was proposed by Denais et al. where the gate voltage was kept constant during stress and measurement to avoid recovery before and after characterization [6]. The drain current was measured periodically to monitor degradation. The underlying assumption in sensor-based techniques is that the circuits under consideration experience the same process, voltage, temperature, state conditions, and ultimately the same aging effects, as the target circuit. This is not true in real circuits, which can cause significant over/under estimation.

Apart from sensor-based techniques, a range of in-situ techniques have also been proposed to monitor wear that is manifested as increased circuit delays. One type of in-situ techniques is to monitor delays during operation. Agarwal et al. proposed a technique to add aging sensor to a flip-flop [1]. This sensor consists of a stability checker, delay element, and a comparator to measure timing degradation. A technique based on shadow registers and comparators combined with negative skew was also proposed [14]. Built-in self-test may also be used to detect delay degradation. Ahmed et al. devised a technique that converts a datapath in a pipeline into a ring oscillator with the help of a multiplexor during testing to measure the oscillation period [2]. The period is then compared to the initial measurements to measure aging of a path. A self-test technique, which monitors delay degradation under environmental variations was also proposed [15]. This technique selects a target path and reconfigures it into a ring oscillator. The period of the reconfigured oscillator is compared to the pre-aging value for estimating aging-induced delay degradation.

The main drawback of existing online wear state monitoring techniques is that they require additions or changes to the underlying hardware. All of the techniques discussed above have large area and testing overheads. In addition, existing techniques also require secret or difficult-to-find design data for the processor under consideration, not to mention precise measurements to predict the wear. To the best of our knowledge, all existing techniques monitor wear due to NBTI in hardware and not in software.

3. RELIABILITY MODEL

We now briefly describe NBTI, EM, TDDB, SM, and TC, which are presently the most dominant device-level failure mechanisms for ICs.

3.1 NBTI

In this paper, we focus on NBTI, as it has the most dominant long term effect in sub-90 nm CMOS process technologies [16]. NBTI is also the primary parametric failure mechanism in modern ICs [4] and a dominant aging mechanism causing PMOS threshold voltage degradation over time [1] resulting in increased delay. NBTI occurs when a negative bias is applied to a PMOS transistor. It has two phases: stress and recovery. In the stress phase, the holes in the channel weaken the Si-H bonds, which result in the generation of positive interface charges and hydrogen species. During recovery phase, the interface traps are annealed by hydrogen species and thus, the degradation of the threshold voltage V_{th} is partially recovered. According to a comprehensive device-level predictive model covering both static and dynamic NBTI degradation [17, 20], the stress phase can be expressed as

$$\Delta V_{th} = \sqrt{K_v^2 \left(t - t_0\right)^{\frac{1}{2}} + \Delta V_{th0}^2 + \delta_v}, \tag{1}$$

where t and t_0 denote the stress and recovery periods, respectively, V_{th0} is the initial threshold voltage, δ_v is a constant for non H-based mechanisms, and

$$K_v = A \cdot t_{ox} \sqrt{C_{ox}\left(V_{gs} - V_{th}\right)} \cdot \left(1 - \frac{V_{gs}}{V_{gs} - V_{th}}\right) e^{\frac{E_{ox}}{E_0}} e^{\frac{E_a}{\kappa T}}, \tag{2}$$

where A is a technology-dependent constant constant, t_{ox} is the oxide thickness, C_{ox} is the oxide capacitance, V_{gs} is the gate voltage, E_{ox} is the electric field across the gate oxide, E_0 is a technology dependent parameter, E_a is the activation energy of hydrogen species, κ is the Boltzmann constant, and T is the temperature.

For the recovery phase, we have

$$\Delta V_{th} = \left(\Delta V_{th} - \delta_v\right)\left(1 - \sqrt{\eta \frac{(t - t_0)}{t}}\right). \tag{3}$$

where η is the surface charge density.

The delay dependence on the threshold voltage is given by the alpha power law as

$$delay \propto \frac{V_{dd}}{\left(V_{dd} - V_{th}\right)^{\beta}}, \tag{4}$$

where V_{dd} is the supply voltage and β is a fitting parameter.

The above equation gives the delay of a single transistor. The overall circuit delay can be obtained by using a timing

analysis scheme based on the increased delay of each transistor. Finally, the path delay degradation can be calculated by analyzing the paths. NBTI manifests as increased circuit delays which eventually lead to timing failures.

3.2 Other Dominant IC Failure Mechanisms

EM refers to the dislocation of metal atoms caused by momentum imparted by electrical current in wires and vias [9, 3]. TDDB involves the deterioration of the gate oxide layer. Gate current due to hot electrons causes defects in the oxide, which eventually form a low-impedance path and cause the transistor to permanently fail. This effect worsens with the reduction of gate dielectric thickness and non-ideal supply voltage reduction [9, 19]. SM is caused by the directionally biased motion of atoms in metal wires due to mechanical stress caused by thermal mismatch among metal and dielectric materials [9]. TC refers to wear caused by thermal stress resulting from mismatched coefficients of thermal expansion for adjacent material layers; run-time temperature variation results in inelastic deformation, eventually leading to failure [22]. The failure rate for EM, TDDB, SM, and TC can be computed as [9]

$$\lambda = K_1 e^{-\frac{K_2}{T}}, \tag{5}$$

where T is the temperature and K_1 and K_2 are temperature-dependent parameters. All the failure mechanisms discussed in this section strongly depend on temperature and, in some cases, on the amplitudes and frequency of the thermal cycles.

4. METHODOLOGY

We now present our proposed online wear state monitoring methodology, which can be used in any embedded processors that allow the control of dynamic voltage and frequency scaling (DVFS) settings and that are equipped with a hardware watchdog timer (WDT) to permit self-restart to completely automate the process. As will be discussed in Section 5.1, we used Embedded Intel Atom N2600 dual core processors to implement our method. As such, some implementation details given in the next section are processor-specific but the general methodology is applicable to all embedded processors supporting DVFS. In addition, we selected Ubuntu as our operating system since it allows the user to take control of the processor's DVFS from user space, making it possible for us to set the frequency and the voltage at which the processor should run.

An overview of our approach is given in Figure 1. The characterization script is used to detect timing violations and, hence, the wear due to NBTI, that occur due to an increase in circuit delays. The profiler supports the wear state monitoring of other dominant IC failure mechanisms, as will be discussed in Section 4.2.

4.1 NBTI Wear Progression Monitoring

Our NBTI wear state monitoring process is built on the following observations. First, different voltage levels lead to different timing slacks in a given critical path. Second, since a higher temperature leads to a reduction of charge carrier mobility in transistors and increase in interconnect latency, negative timing slacks are more likely to occur at higher temperature. Third, NBTI increases circuit delay, which can lead to timing violations. Fourth, insufficient timing slacks lead to more observable crashes when running programs.

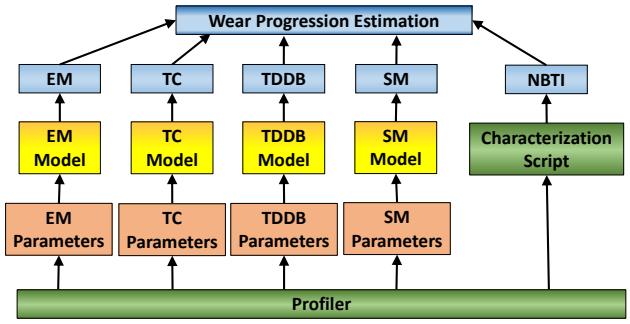


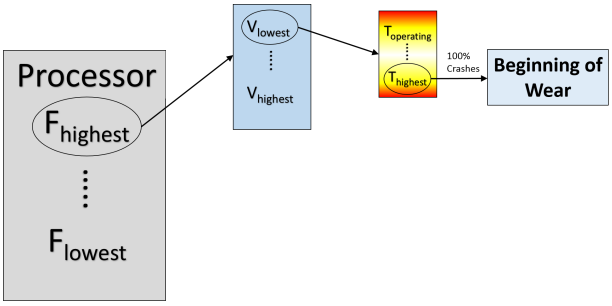
Figure 1: Overview of the proposed online wear monitoring process.

The frequency of a processor is determined by the delay at the slowest corner. This slowest corner corresponds to the lowest voltage level and the highest temperature at which the processor can still function within its required timing constraints. Slow corners are more vulnerable to timing faults as they have less timing slack when compared to voltage and frequency points operating at lower temperatures. Hence processors are more likely to fail due to NBTI effects when operating at these slow corners. Note that while the change in the threshold voltage due to an increase in temperature may improve performance, the reduction of charge carrier mobility in transistors dominates, making the circuit slower overall.

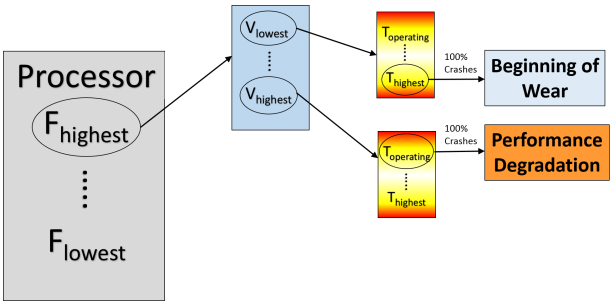
We propose performing crash tests at specific VFT points to monitor, and potentially predict, how wear due to NBTI progresses over time. Figure 2 provides an illustration of the relationship between the VFT points and the wear state of a processor due to NBTI, which is a fundamental observation in which this work is based on. For a given frequency level F, there is a discrete number of valid voltage levels ranging from V_{lowest} to $V_{highest}$, which may be different from one frequency level to another and from one processor to another. The values for $T_{ambient}$ and $T_{highest}$ denote the processor temperature when operating at the lowest voltage and frequency pair and the highest temperature the processor can run without triggering hardware throttling, respectively.

We now explain the intuition behind our approach. If there are 100% crashes at the slowest corner (V_{lowest}, $T_{highest}$, $F_{highest}$) (Figure 2(a)), then the processor has begun to show signs of wear due to NBTI. As another example, if the processor experiences 100% crashes at ($V_{highest}$, $T_{ambient}$, $F_{highest}$) (Figure 2(b)), the processor's performance degradation has become more noticeable. Finally, if the processor cannot function at ($V_{highest}$, $T_{ambient}$, F_{lowest}) (Figure 2(c)), the processor has reached the end of its life. The change in crash conditions shows the progression of wear over time.

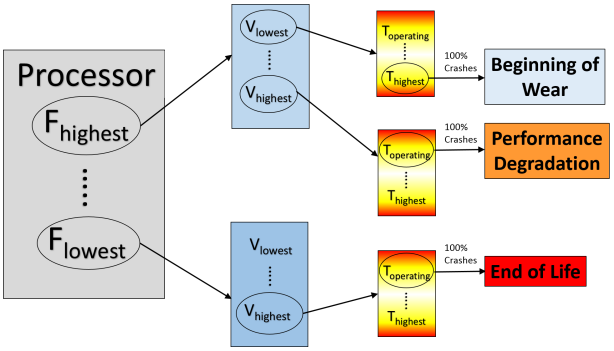
To monitor in real-time the wear progression of a processor, we construct a characterization program, i.e., the characterization script in Figure 1, to test the VFT points. Due to process variations, VFT points may be different from one processor to another. Therefore, VFT points are built individually for a given processor. This is a one-time process. All the voltage values that allow the processor to run without crashing with the initial temperature being a controllable ambient temperature are considered valid. The characterization program executes a series of tests at specific VFT



(a) If there are 100% crashes at the slowest corner, (V_{lowest}, $T_{highest}$, $F_{highest}$), then the processor has begun to show signs of wear.



(b) If the processor experiences 100% crashes at ($V_{highest}$, $T_{ambient}$, $F_{highest}$), the processor's performance degradation becomes more noticeable.



(c) Finally, if the processor cannot function at ($V_{highest}$, $T_{ambient}$, F_{lowest}), the processor has reached the end of its life.

Figure 2: Relationship between VFT points and the wear state of a processor over time.

points and collects the corresponding number of crashes for later analysis.

An overview of our characterization software is provided in Figure 3. The objective of the characterization process for a given processor is to perform periodic crash tests during a time period when the processor is not in use. The characterization of processor wear is performed by a user space application. An init script is scheduled to run at specific times using `gnome-schedule`, a graphical user interface to manage `crontab`. It is also run as a startup task to start several bash scripts that are part of the monitoring process. There are three main modules in our wear state monitoring software: profiler, watchdog timer program, and characterization script. We now describe each module and its purpose in detail.

4.1.1 *Profiler*

To facilitate the data collection and subsequent analysis, a profiler script is used to periodically record the temperature, voltage, and frequency of each core. The temperatures of the cores are read through the `coretemp` driver which in turn reads the temperatures from the on-die digital thermal sensors. The profiler also records the frequency and voltage at which the processor is operating. The current performance state (p-state) value of the processor can be obtained from the model specific register (MSR) located at address `0x198, IA32_PERF_STATUS`. Bits [15:8] of the MSR indicate the frequency while bits [7:0] hold the voltage. The current VFT point is written to a file and the profiler wakes up again later to repeat the process. Note that while the specific processor used in our experiments has a single frequency and voltage setting for all the cores, the script can be readily used to collect the frequency and voltage information of each individual core.

4.1.2 *Watchdog Timer Program*

A challenge in performing wear state monitoring arises when a processor crashes during testing. One solution is to rely on human intervention to restart the system every time it crashes. Clearly, this is not a viable solution, especially for embedded systems that are deployed in remote locations and where the testing times may be at odd hours at night. To fully automate the wear state monitoring process, we propose using a WDT to automatically restart the system once the latter has become unresponsive for a certain time interval.

The program to control the WDT is written is C and is activated only when the characterization script (see the next section) is active. The WDT is set for a fixed time interval. Once the timer expires, the watchdog timer program reads a file to infer if the characterization script is still alive. If the execution of the script has been frozen, it can be concluded that the processor has crashed as a result of timing violations. In such a case, the watchdog timer restarts the system. Otherwise, it deletes the file for the characterization script to write to again and goes to sleep.

4.1.3 *Characterization Script*

The characterization script is the backbone of our online wear state monitoring tool. It tests each VFT point for a given processor and runs periodically. During characterization, the temperature is first adjusted to the desired temperature point by increasing or decreasing the load on the processor with the help of `burnMMX`, a program designed to load x86 CPUs as heavily as possible for system testing. Once the processor has reached the desired temperature, the voltage and frequency pair to be tested for are written to `MSR 0x199, IA32_PERF_CTL`. Bits [15:0] of the MSR represent the target p-state value, with bits [15:8] indicating the frequency and bits [7:0] indicating the voltage. MSR Tools [8] are used to read and write data to the MSR. To ensure that the current p-state is the same as the target p-state, bits [15:0] of `IA32_PERF_STATUS` is read. That is, the value written in `IA32_PERF_CTL` will appear in `IA32_PERF_STATUS` only if the scaling governor is changed from on-demand to user space.

As described in the previous subsection, the characterization script writes to a file for the watchdog timer program to read and ascertain that the test is running properly and that the processor has not crashed. Each test run is inde-

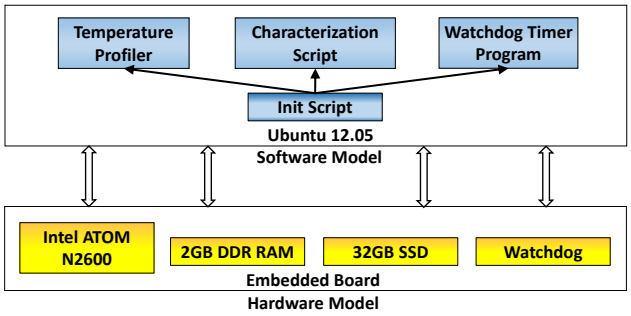


Figure 3: A hardware/software architecture for the proposed wear state monitoring tool.

pendent of the previous one in that it is not influenced by it. Specifically, since a test may raise the temperature of a processor and change the state of the operating system, the processor is restarted and cooled down to the ambient temperature before the next test is run. A processor is said to have crashed if it is frozen and has been restarted by the watchdog timer after a VFT point is set.

Each VFT point is tested n times where n is user-adjustable. We selected 10 to achieve a balance between having the crash data being statistically significant and keeping the overhead associated with the tool acceptable. If there are no crashes at a VFT point, then the wear progression due to NBTI has not yet affected the processor when operating at that VFT point. As a result, that VFT point can still be reliably used. If the processor consistently crashes at a certain VFT point, said point can no longer be used for reliable operation. Otherwise, reliable execution of applications at that VFT point is not fully guaranteed and the VFT point in question may need to be avoided. In any case, knowledge on the VFT points can be shared with the users or system designers so that they can make informed decisions when selecting which VFT point to run their applications.

A pseudocode of the characterization program for online wear state monitoring is shown in Algorithm 1. There are several inputs. First, we have a set of discrete frequencies sorted in decreasing order, denoted by F. For each frequency F_i, we have a set of discrete voltage levels V_i, sorted in increasing order. The temperature points T whose values are between $[T_{ambient}, T_{highest}]$ are sorted in decreasing order. As previously stated, $T_{ambient}$ and $T_{highest}$ refer to the ambient and highest temperature values for the processor under consideration, respectively. All the inputs can be determined offline; this is a one-time process. For each frequency F_i, we set the processor temperature at $T_{ambient}$ and keep decreasing the voltage. The lowest voltage at which the processor does not crash is recorded as V_{lowest} for the corresponding F_i.

The test starts with the highest frequency and decreases it until the lowest frequency is reached (Line 2). For each frequency, the voltage to be tested for progresses from the lowest voltage level to the highest voltage level (Line 3). For a given voltage-frequency pair, the temperature to be tested against starts from $T_{highest}$, which represents the slowest corner, down to $T_{ambient}$ (Line 7). The test starts with $(F_{highest}, T_{highest}, T_{highest})$ because it has the shortest critical path and timing violations are more likely to occur. The

Algorithm 1 Characterization(F, V, T, $T_{highest}$, $T_{ambient}$)

1: $crash\_count_{i,j,k} \leftarrow 0$, $\forall i = 1, \ldots, |F|$, $\forall j = 1, \ldots |V_i|$, $\forall k = 1, \ldots, |T|$
2: **for** each $F_i \in F$ **do** // Frequency levels are sorted in decreasing order
3: **for** each $V_{i,j} \in V_i$ **do** // V_i is the set of increasing voltage levels that can operate at F_i
4: **if** $crash\_count_{i,j-1,|T|} = 0$ **then**
5: **break**
6: **else**
7: **for** each $T_k \in T$ **do** // Temperature points are sorted in decreasing order
8: set processor temperature to T_k
9: $runs \leftarrow 0$
10: **if** $crash\_count_{i,j,k-1} > 0$ **then**
11: **while** $runs < 10$ **do**
12: set processor frequency to F_i
13: set processor voltage to $V_{i,j}$
14: delay($wait\_period$)
15: **if** processor crashes **then**
16: $crash\_count_{i,j,k} \leftarrow crash\_count_{i,j,k} + 1$
17: **end if**
18: $run \leftarrow run + 1$
19: restart processor for next test
20: set processor temperature to $T_{ambient}$
21: **end while**
22: **end if**
23: **end for**
24: **end if**
25: **end for**
26: **end for**

test progresses through VFT points with critical paths in the increasing order. The temperature granularity is a user-adjustable parameter. Given a VFT point, crash tests are carried out only if crashes had occurred for the VF pair at a higher temperature (Line 10). Otherwise, there is no need to continue testing this VF pair since crashes are not likely to occur. Tests for crashes start after a certain time interval once the processor voltage and frequency values have been set (Lines 12–14). We experimented with different values for the wait period (Line 14) and found that, in our case, if a processor will crash, it will do so within three minutes. Hence, the waiting period is set to 3 minutes.

For every voltage level to be tested, we first determine whether there were crashes when using the previous (lower) voltage level at the highest temperature (Line 4). If so, the testing process continues. Otherwise, testing is completed. Since the system's timing slack for a higher voltage point is greater than that for a lower voltage point, there is no need to check for crashes at a higher voltage level when there are none at the lower voltage level. To obtain the distribution of the crashes, we collect data at every VFT point until no crash is observed. In practice, a binary search-based approach may be adopted to avoid testing all VFT points if the distribution of the crashes is known. For clarity, we omitted some checkpoints from the pseudocode in Algorithm 1.

The crash counts can directly be used to monitor the wear progression due to NBTI of a processor over time. Specifically, if the crash count for a VFT point is consistently 10, the use of that VFT point should be avoided. In addition, we can monitor the change in the crash count for a VFT

point over time to determine how close a processor is to failing. The characterization process is fairly lightweight in that each test does not take much time to run, as will be discussed in Section 5.3.

4.2 Wear Progression Monitoring of Other IC-Dominant Failure Mechanisms

Failures due to some of the mechanisms discussed in Section 3.2 cannot be predicted using wear state sensors. We would like to point out that our online wear monitoring tool naturally supports wear state monitoring for other dominant IC failure mechanisms such as EM, TDDB, SM, and TC, provided that model-specific parameters are known. That is, since wear due to these failure mechanisms mainly depends on temperature, we can trivially use the profiler (Section 4.1.1) to obtain the thermal profile for a given processor over a certain time interval. This thermal profile, along with the operating voltage and frequency values, which are also collected by the profiler, can be used as inputs to a system-level reliability modeling and analysis tool [10, 21]. Such a tool would be able to predict wear progression as long as the constants in equation (5) are known.

5. EXPERIMENTS

We now present our experimental setup, describe how we stressed the processors for accelerated aging, and discuss the results obtained from our wear state monitoring tool.

5.1 Setup

We acquired five off-the-shelf industrial standard embedded boards. However, we only discuss the setup and data on three boards, as during the course of our experiments, we ran into both hardware and operating system issues with two of the boards. Each board is a single board computer containing an Embedded Intel Atom N2600 dual core processor with a maximum frequency of 1.6 GHz and is manufactured in the 32 nm silicon-on-insulator process technology. The processor has Enhanced Intel SpeedStep Technology, Intel's proprietary DVFS technology, enabled. There are 6 unique frequencies and each frequency has a discrete set of voltages it can support (Figure 4). While the processor voltage can be regulated at a granularity of 0.005V, the board imposes a granularity constraint of 0.01V. As can be seen in Figure 4, for a given frequency, the supported voltage range may be different from one processor to another due to process variations.

Each board has 2 GB DDR RAM, a 32 GB SSD and a WDT. We installed Ubuntu 12.05 and our online wear state monitoring tool on all the boards. Each board is connected to an independent 12 V power source, which is in turn connected to a power strip with a multi-system timer. The WDT and the power strip with timer are used to completely automate the characterization of the processor.

We conducted a one-time offline process to determine the values for $T_{ambient}$ and $T_{highest}$. These data are shown in Table 1. In order to get the processor temperature to be very close to the throttling temperature, we removed the heatsink on each board. Consequently, the ambient temperature is higher. Note that the heatsink removal was part of the stress tests we ran to accelerate the wear on the processors and is not a requirement imposed by our wear state monitoring tool. In our tests, the temperature step size in Algorithm 1 is 5 °C and the wait period was set to 3 minutes.

	Frequency (GHz)	Voltage (V)	Number of VF Points
Board 1	1.6	0.987–0.887	12
	1.4	0.987–0.837	16
	1.2	0.987–0.797	20
	1.0	0.987–0.777	22
	0.8	0.987–0.777	22
	0.6	0.987–0.777	22
Board 2	1.6	0.987–0.886	12
	1.4	0.996–0.846	16
	1.2	0.996–0.806	20
	1.0	0.996–0.776	23
	0.8	0.996–0.776	23
	0.6	0.996–0.776	23
Board 3	1.6	0.987–0.874	12
	1.4	0.984–0.834	16
	1.2	0.984–0.774	20
	1.0	0.984–0.774	22
	0.8	0.984–0.774	22
	0.6	0.984–0.774	22

Figure 4: Voltage and frequency information for the Intel Atom N2600 dual core processor. The voltage level resolution is 0.01 V.

5.2 CPU Stress Script

To accelerate the wear of the boards for experimental purposes, we made use of a CPU stress script, which will not be run on real systems. In this work, we ran the CPU stress script during the day and performed the characterization process at night. The CPU stress script has two patterns of stress. The first stress pattern is designed to keep a processor temperature close to the throttling temperature. It keeps running the processor at $T_{highest}$ to accelerate wear due to NBTI (as well as EM, TDDB, and SM). In the second stress pattern, the processor cycles between $T_{ambient}$ and $T_{highest}$ as often as possible to accelerate wear due to TC if desired.

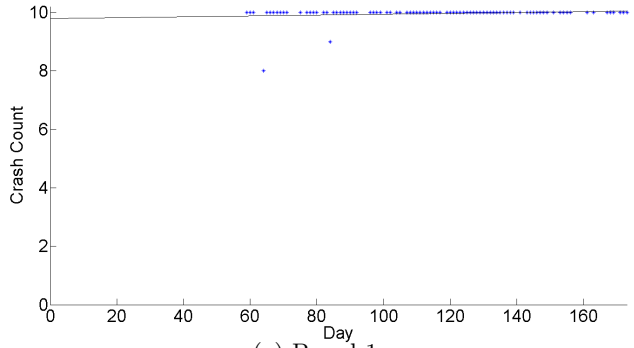
5.3 Results

The experiments were conducted over the course of 160 days. Each test takes about 5 minutes. The total time taken to perform wear monitoring will depend on the number of VFT points as well as the number of test runs. We set the number of test runs to 10 in this work. To illustrate the wear progression of processors due to NBTI over time, we plot the number of crashes as a function of time and include the corresponding least squared curve fit line. Data are shown in Figure 5 when the tests were run at ($F_{highest}$, V_{lowest}, $T = T_{highest}$) for the respective boards. This setting corresponds to the slowest corner of a given processor and is expected to show signs of wear early on. Although the number of crashes over time does not monotonically increase, a trend can be observed. In addition, even with noise, it can be seen from the plots the point beyond which a user may no longer wish to run the processor at a particular VFT point, e.g., around Day 60 for Board 1 in Figure 5. The least squared error, i.e., best fit, line shows a steadily increasing trend in terms of crash counts over time.

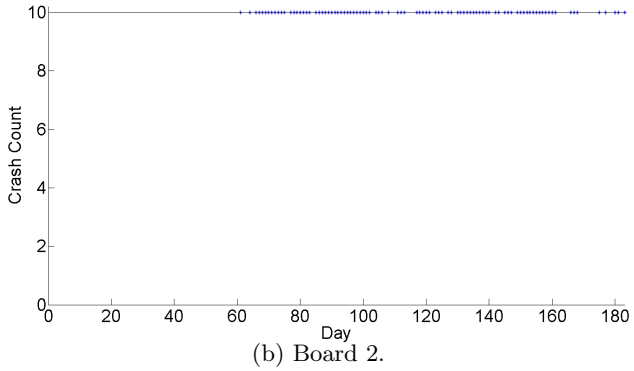
It is not surprising that the slowest corner of a processor would expose wear due to NBTI over time. We now present experimental data for ($F_{highest}$, V_{lowest}, $T = T_{highest} - 10$) for the respective processors in Figure 6. The number of crash counts over time is still obvious, though not as dramatic as in the previous case. At around Day 100, it can be seen that Boards 1 and 3 can still operate at the stated VFT points but that timing violations are more likely to occur.

We expect that if the experiments were run for a period longer than 160 days, the trends would approach those seen in Figure 5.

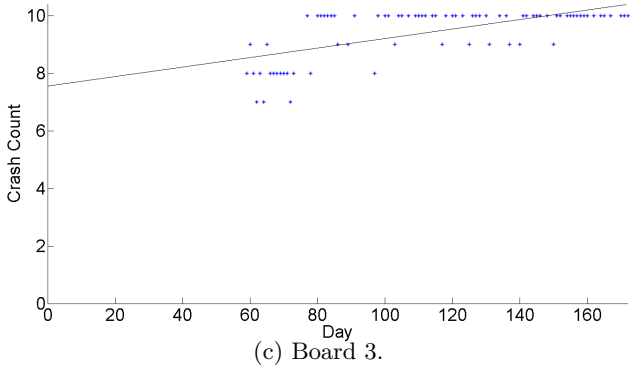
Figure 7 shows the data for when a processor begins to show an early sign of wear at a VFT point. Here, the frequency and voltage settings are the same as before, but the initial temperature of the processor was set to its ambient temperature. Wear progression at this stage is easy to observe. From the data, we can also confirm that different boards start out at different initial wear state and that, due to process variations, did not age at the same rate.



(a) Board 1.



(b) Board 2.



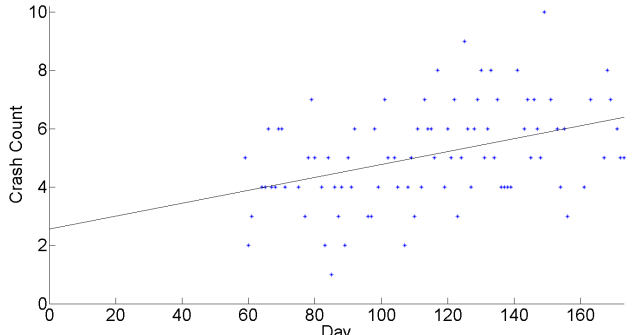
(c) Board 3.

Figure 5: Crash counts over time for the different boards at ($F_{highest}$, V_{lowest}, $\mathbf{T = T}_{highest}$). Even with noise, the increase in wear due to NBTI over time can be observed.

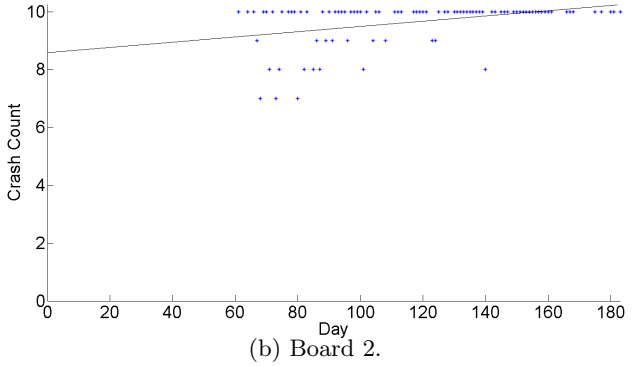
Since we have data on the crash counts for different wear stages, i.e, at different temperature points for a given VF pair, we can plot the distribution of the number of crashes per crash point. A crash point refers to the number of crashes for each set of 10 test runs. For example, if the crash point is 2, the crash count was 2 out of 10 test runs. The results are shown in Figure 8 for Board 1. The y-axis denotes the number of crashes that occurs at each crash point over time. The distribution of crash probabilities appear to

Table 1: Temperature Constants.

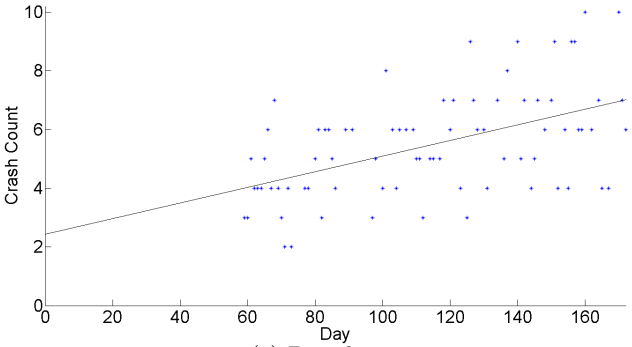
Board	$T_{ambient}$ (°C)	$T_{highest}$ (°C)	Temperature Range (°C)
Board 1	76	91	15
Board 2	75	95	20
Board 3	77	92	15



(a) Board 1.

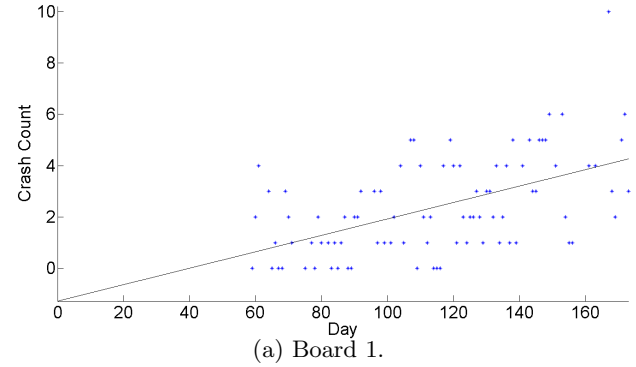


(b) Board 2.

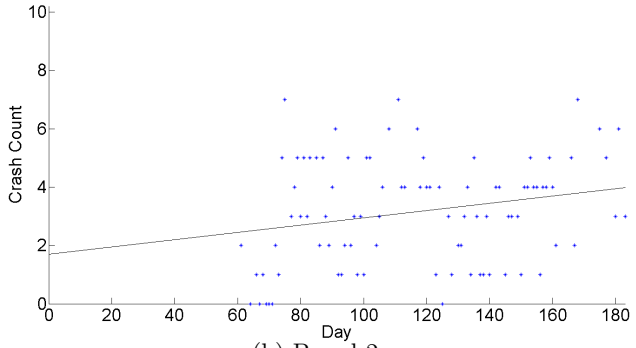


(c) Board 3.

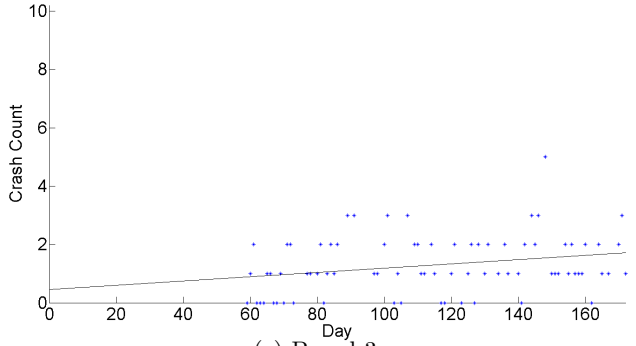
Figure 6: Crash counts over time for the different boards at ($F_{highest}$, V_{lowest}, $\mathbf{T = T}_{highest} - \mathbf{10}$). The wear progression over time is still obvious, though not as dramatic as in the previous case. The least squared error line shows a steadily increasing trend in terms of crash counts over time.



(a) Board 1.

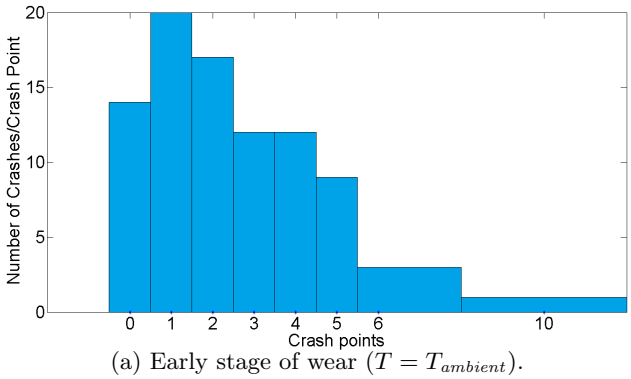


(b) Board 2.

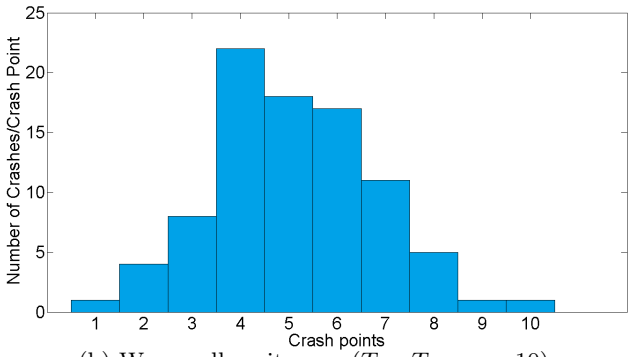


(c) Board 3.

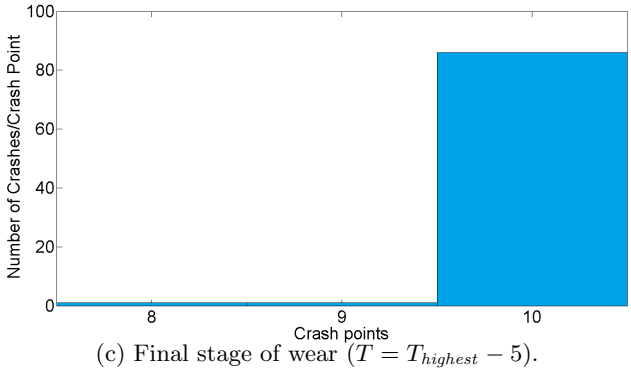
Figure 7: Crash counts over time for the different boards at ($F_{highest}$, V_{lowest}, $\mathbf{T = T}_{ambient}$). The processor begins to show early signs of wear.



(a) Early stage of wear ($T = T_{ambient}$).



(b) Wear well on its way ($T = T_{highest} - 10$).



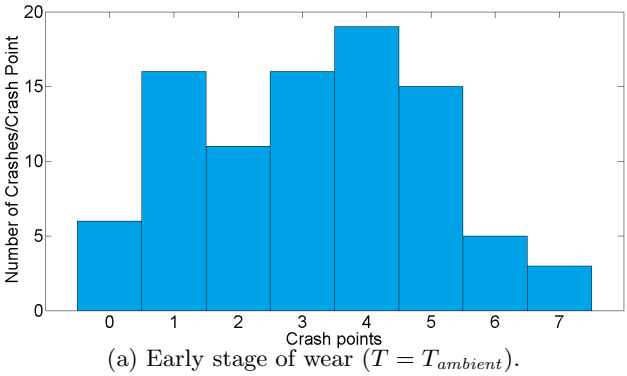
(c) Final stage of wear ($T = T_{highest} - 5$).

Figure 8: Distribution of the number of crashes per crash point for different wear stages for Board 1 when operating at the highest frequency and lowest valid voltage level. The distribution of crash probabilities appear to have a truncated Gaussian distribution. In the early stage of wear, crashes usually occur at a mean of 10% - 30%. With wear well on its way, the mean shifts to the center at 50%-60%. At this wear stage, it is unreliable for a processor to be operating at the corresponding VFT point. Finally, when a VFT point can no longer be used, the mean shifts to 100%.

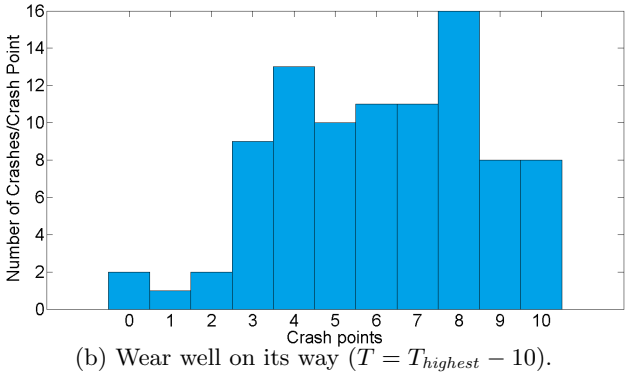
have a truncated Gaussian distribution. In the early stage of wear (Figure 8(a)), crashes usually occur at a mean at 10% - 30%. With wear well on its way (Figure 8(b)), the mean shifts to the center at 50%-60%. At this wear stage, it is unreliable for a processor to be operating at the corresponding VFT point. Finally, when a VFT point can no longer be used (Figure 8(c)), the mean shifts to 100%. We believe that this understanding of how wear progresses over time will help in the design of reliable real-time embedded systems. For completeness and to show that the trend we observe is a general one, we include the distributions for different wear stages for Boards 2 and 3 in Figures 9 and 10, respectively.

6. CONCLUSIONS & FUTURE WORK

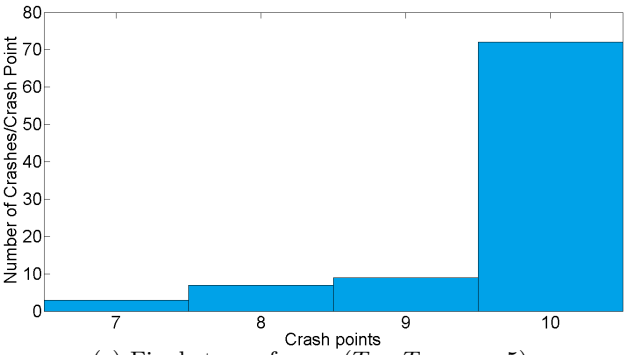
We presented an online, software-only, methodology for monitoring wear due to NBTI of off-the-shelf embedded processors. The wear progression of NBTI, which causes increased circuit delays, can be detected over time by per-



(a) Early stage of wear ($T = T_{ambient}$).

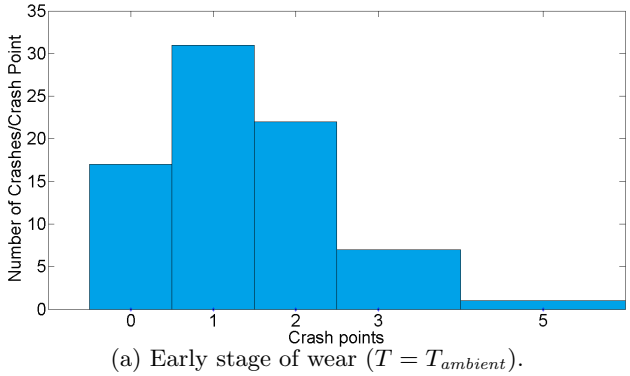


(b) Wear well on its way ($T = T_{highest} - 10$).

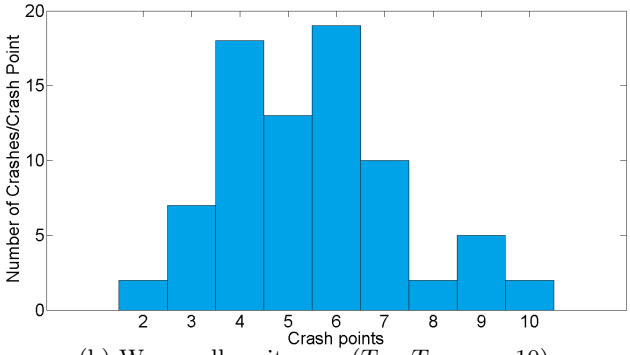


(c) Final stage of wear ($T = T_{highest} - 5$).

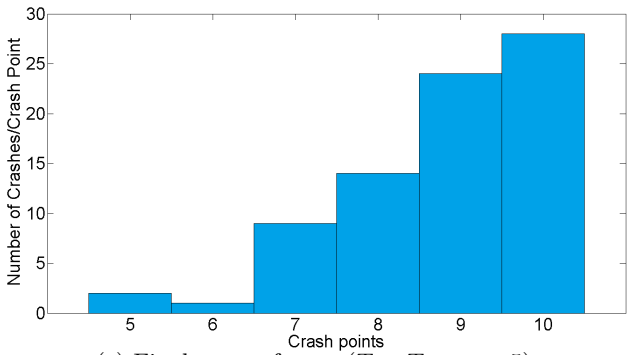
Figure 9: Distribution of the number of crashes per crash point for different wear stages for Board 2 when operating at the highest frequency and lowest valid voltage level.



(a) Early stage of wear ($T = T_{ambient}$).



(b) Wear well on its way ($T = T_{highest} - 10$).



(c) Final stage of wear ($T = T_{highest} - 5$).

Figure 10: Distribution of the number of crashes per crash point for different wear stages for Board 3 when operating at the highest frequency and lowest valid voltage level. The trend shown here are similar to the ones in Figures 8 and 9.

forming crash tests at selective VFT points. The proposed methodology is able to monitor the wear progression due to EM, SM, TDDB, and TC as well if and when model-specific parameters are available. The main advantage of our approach over wear state sensors are that (i) it does not require modifications to hardware and (ii) it can help to monitor wear due to failure mechanisms that may not be detected using sensors. Our online monitoring tool only requires a watchdog timer and software controllable DVFS settings. It is our vision that the said tool can aid in the design of reliable and predictable real-time embedded systems by providing feedback at runtime so that the use of unreliable VFT points can be avoided.

This work can be extended in several directions. First, we plan on leveraging our wear state monitoring technique to include wear and, ultimately, lifetime predictions. Second, it may be useful to adapt our methodology to monitor the wear progression of server computers. Third, a mechanism to determine reliability-related parameters for EM, SM, TDDB, and TC would be of great value.

7. ACKNOWLEDGEMENTS

This work was supported in part by NSF under awards CNS-1319904, CNS-1319718, and CNS-1319784 and in part by Sandia National Laboratory.

8. REFERENCES

[1] M. Agarwal, B. Paul, M. Zhang, and S. Mitra. Circuit failure prediction and its application to transistor aging. In *Proc. VLSI Test Symp.*, pages 277–286, May 2007.

[2] F. Ahmed and L. Milor. Built-in self test circuit for delay degradation detection. In *Proc. Int. Conf. Design Circuits Integrated System*, pages 65–70, Nov. 2009.

[3] J. R. Black. Electromigration–a brief survey and some recent results. *IEEE Trans. Electron Devices*, 16(4):338–347, Apr. 1969.

[4] A. C. Cabe, Z. Qi, S. N. Wooters, T. N. Blalock, and M. R. Stan. Small embeddable NBTI sensors SENS for tracking on-chip performance decay. In *Proc. Int. Symp.Quality Electronic Design*, pages 1–6. IEEE, Mar. 2009.

[5] S. Chakravarthi, A. Krishnan, V. Reddy, C. Machala, and S. Krishnan. A comprehensive framework for predictive modeling of negative bias temperature instability. In *Proc. Int. Reliability Physics Symp.*, pages 273–282, Apr. 2004.

[6] M. Denais, C. Parthasarathy, G. Ribes, Y. Rey-Tauriac, N. Revil, A. Bravaix, V. Huard, and F. Perrier. On-the-fly characterization of NBTI in ultra-thin gate oxide PMOSFET's. In *Proc. Int. Electron Devices Meeting*, pages 109–112, Dec. 2004.

[7] A. Ghosh, R. Brown, R. Rao, and C.-T. Chuang. A precise negative bias temperature instability sensor using slew-rate monitor circuitry. In *Prof. Int. Symp. Circuits and Systems*, pages 381–384, May 2009.

[8] Intel Open Source. msr-tools. `https://01.org/msr-tools`.

[9] Failure mechanisms and models for semiconductor devices. Technical report, Joint Electron Device Engineering Council, Aug. 2003. JEP 122-B.

[10] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge. Reliability modeling and management in dynamic microprocessor-based systems. In *Proc. Design Automation Conf.*, pages 1057–1060, July 2006.

[11] J. Keane, T.-H. Kim, and C. Kim. An on-chip NBTI sensor for measuring PMOS threshold voltage degradation. *IEEE Trans. VLSI Systems*, 18(6):947–956, June 2010.

[12] T.-H. Kim, R. Persaud, and C. Kim. Silicon odometer: An on-chip reliability monitor for measuring frequency degradation of digital circuits. *IEEE J. Solid-State Circuits*, 43(4):874–880, Apr. 2008.

[13] H. Kufluoglu and M. Alam. A generalized reaction–diffusion model with explicit H–H_2 dynamics for negative-bias temperature-instability (NBTI) degradation. *IEEE Trans. Electron Devices*, 54(5):1101–1107, May 2007.

[14] J. Li and J. Lach. Negative-skewed shadow registers for at-speed delay variation characterization. In *Proc. Int. Conf. Computer Design*, pages 354–359, Oct. 2007.

[15] J. Li and M. Seok. Robust and in-situ self-testing technique for monitoring device aging effects in pipeline circuits. In *Proc. Design Automation Conf.*, pages 1–6, June 2014.

[16] J. Pachito, C. V. Martins, J. C. Vazquez, B. Jacinto, I. Teixeira, J. Teixeira, V. Champac, J. Semiao, and M. B. Santos. Aging-aware power or frequency tuning with predictive fault detection. *IEEE Design & Test of Computers*, 29(5):27–36, June 2012.

[17] K. Saluja, S. Vijayakumar, W. Sootkaneung, and X. Yang. NBTI degradation: A problem or a scare? In *Proc. Int. Conf. VLSI Design*, pages 137–142, Jan. 2008.

[18] P. Singh, E. Karl, D. Blaauw, and D. Sylvester. Compact degradation sensors for monitoring NBTI and oxide degradation. *IEEE Trans. VLSI Systems*, 20(9):1645–1655, Sept. 2012.

[19] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. Exploiting structural duplication for lifetime reliability enhancement. In *Proc. Int. Symp. Computer Architecture*, pages 520–531, June 2005.

[20] R. Vattikonda, W. Wang, and Y. Cao. Modeling and minimization of PMOS NBTI effect for robust nanometer design. In *Proc. Design Automation Conf.*, pages 1047–1052, July 2006.

[21] Y. Xiang, T. Chantem, R. P. Dick, X. S. Hu, and L. Shang. System-level reliability modeling for MPSoCs. In *Proc. Int. Conf. Hardware/Software Codesign and System Synthesis*, pages 297–306, Oct. 2010.

[22] Y. Zhang and K. Chakrabarty. Task feasibility analysis and dynamic voltage scaling in fault-tolerant real-time embedded systems. In *Proc. Design, Automation & Test in Europe Conf.*, pages 1170–1175, Feb. 2004.