Self-calibrating Online Wearout Detection

Jason Blome Shuguang Feng Shantanu Gupta Scott Mahlke

Advanced Computer Architecture Laboratory
University of Michigan
Ann Arbor, MI 48109
{jblome, shoe, shangupt, mahlke}@umich.edu

ABSTRACT

Technology scaling, characterized by decreasing feature size, thinning gate oxide, and non-ideal voltage scaling, will become a major hindrance to microprocessor reliability in future technology generations. Physical analysis of device failure mechanisms has shown that most wearout mechanisms projected to plague future technology generations are progressive, meaning that the circuit-level effects of wearout develop and intensify with age over the lifetime of the chip. This work leverages the progression of wearout over time in order to present a low-cost hardware structure that identifies increasing propagation delay, which is symptomatic of many forms of wearout, to accurately forecast the failure of microarchitectural structures. To motivate the use of this predictive technique, an HSPICE analysis of the effects of one particular failure mechanism, gate oxide breakdown, on gates from a standard cell library characterized for a 90 nm process is presented. This gate-level analysis is then used to demonstrate the aggregate change in output delay of high-level structures within a synthesized Verilog model of an embedded microprocessor core. Leveraging this analysis, a self-calibrating hardware structure for conducting statistical analysis of output delay is presented and its efficacy in predicting the failure of a variety of structures within the microprocessor core is evaluated.

1. INTRODUCTION

Traditionally, microprocessors have been designed with worst case operating conditions in mind. To this end, manufacturers have employed burn in, guard bands, and speed binning to ensure that processors will meet a predefined lifetime qualification, or mean time to failure (MTTF). However, projections of current technology trends indicate that these techniques are unlikely to satisfy reliability requirements in future technology generations [14]. As CMOS feature sizes scale to smaller dimensions, the inability of operating voltage to scale accordingly results in dramatic increases in power and current density. Areas of high power density increase local temperatures leading to hot spots on the chip [29]. Since most wearout mechanisms, such as gate oxide breakdown (OBD), negative bias temperature instability (NBTI), electromigration (EM), and hot carrier injection (HCI), are highly dependent on temperature, power, and current density, the occurrence of wearout-induced failures will become increasingly common in future technology generations [2].

Though the reliability of individual devices is projected to decrease, consumer expectations regarding system reliability are only likely to increase. For example, some business customers have reported downtime costs of more than $1 million per hour [6]. Further, a recent poll conducted by Gartner Research demonstrated that more than 84% of organizations rely on systems that are over five years old, and more than 50% use machines that are over ten years old [1]. Given the requisite long life expectancies of systems in the field and the high costs of in-field replacement, any technique for mitigating the amount of downtime experienced due to failed systems will prove invaluable to businesses.

In order to maintain availability in the presence of potentially unreliable components, architects and circuit designers have historically employed either error detection or failure prediction mechanisms. Error detection is used to identify failed or failing components by locating (potentially transient) pieces of incorrect state within the system. Once an error is detected, the problem is diagnosed and corrective actions may be taken. The second approach, failure prediction, supplies the system with a failure forecast allowing it to take preventative measures to avoid, or at least minimize, the impact of expected device failures.

Historically, high-end server systems have relied on error detection to provide a high degree of system reliability. Error detection is typically implemented through coarse grain replication. This replication can be conducted either in space through the use of replicated hardware [32, 12], or in time by way of redundant computation [27, 24, 22, 37, 30, 18, 25, 23]. The use of redundant hardware is costly in terms of both power and area and does not significantly increase the lifetime of the processor without additional cold-spare devices. Detection through redundancy in time is potentially less expensive but is generally limited to transient error detection unless redundant hardware is readily available.

Failure prediction techniques are typically less costly to implement, but they also face a number of challenges. One traditional approach to failure prediction is the use of canary circuits [4], designed to fail in advance of the circuits they are charged with protecting, providing an early indication that important processor structures are nearing their end of life. Canary circuits are an efficient and generic means to predict failure. However, there are a number of sensitive issues that must be addressed to deploy them effectively. For instance, the placement of these circuits is extremely important for accurate prediction, because the canary must be subjected to the same operating conditions as the circuit it is designed to monitor.

Another technique for failure prediction is the use of timing sensors that detect when circuit latency is increasing over time or has surpassed some predefined threshold [17, 13, 5]. The work presented here extends upon [13] which presented the idea of failure prediction using timing analysis and identifying degrading performance as a symptom of wearout in semiconductor devices.

Recent work by Srinivasan [34] proposes a predictive technique that monitors the dynamic activity and temperature of structures within a microprocessor in order to calculate their predicted time to failure based on analytical models. This system can then be used to swap in cold-spare structures based on these predictions. This work pioneered the idea of dynamically trading performance for reliability in order to meet a predefined lifetime qualification. Although this technique may be used to identify structures that are likely to fail in the near future, it relies on accurate analytical device wearout models and a narrow probability density function for effective predictions.

Research into the physical effects of wearout on circuits has shown that many wearout mechanisms for silicon devices are progressive over time. Further, many of these wearout mechanisms, such as EM, OBD, HCI, and NBTI, have been shown to have a negative impact on device performance [7, 20, 41, 16]. For example, a device subject to hot carrier injection (HCI) will experience drive current degradation, which leads to a decrease in switching frequency [7]. The recognition of progressive performance degradation as a precursor to wearout-induced failures creates a unique opportunity for predictive measures, which can forecast failures by dynamically analyzing the timing of logic in situ.

The work presented here proposes an online technique that detects the performance degradation caused by wearout over time in order to anticipate failures. Rather than aggressively deploying duplicate fault-checking structures or relying on analytical wearout models, an early warning system is presented that identifies the performance degradation symptomatic of wearout. As a case study, and to derive an accurate performance degradation model for subsequent simulations, detailed HSPICE simulations were performed to determine the impact of one particular wearout mechanism, OBD, on logic gates within a microprocessor core. Research of other progressive wearout mechanisms such as HCI and EM, indicates that similar effects are likely to be observed as a result of these phenomenon.

The results of this analysis are used to motivate the design of an online latency sampling unit, dubbed the wearout detection unit (WDU). The WDU is capable of measuring the signal propagation latencies for signals within microprocessor logic. This information is then sampled and filtered by a statistical analysis mechanism that accounts for anomalies in the sample stream (caused by phenomenon such as clock jitter, and power and temperature fluctuations). In this way, the WDU is able to identify significant changes in the latency profile for a given structure and predict a device failure. Online statistical analysis allows the WDU to be self-calibrating, adapting to each structure that it monitors, making it generic enough to be reused for a variety of microarchitectural components.

Traditional studies of wearout mechanisms have focused primarily on their effects on transistor and circuit level performance, without analyzing the microarchitectural impact. To the best of the authors' knowledge, the experiments presented in this work are the first such attempt in this direction. The contributions of this paper include:

- An HSPICE-based characterization of OBD-induced wearout
- A microarchitectural analysis of the performance impact of OBD on microprocessor logic
- A detailed simulation infrastructure for modeling the impact of wearout on an embedded processor core
- A self-calibrating WDU capable of monitoring path latencies
- A demonstration of how the WDU can be deployed to extend processor lifetime

2. DEVICE-LEVEL WEAROUT ANALYSIS

Though many wearout mechanisms have been shown to progressively degrade performance as transistors age [7, 41, 16], as a case study, this work focuses on the effects of one particular mechanism, gate oxide breakdown (OBD), to demonstrate how performance degradation at the device level can affect processor performance at the microarchitectural level. Due to the lack of microarchitectural models for the progressive effects of wearout, it was necessary to first model the effects at the circuit level in order to abstract them up to the microarchitecture. The results of the modeling and abstraction are presented within this section. While this section is useful in understanding the nature of progressive wearout, readers unfamiliar with device physics may want to simply note the high-level abstraction of OBD effects presented in Figure 1 and move on to section 3.

The remainder of this section describes the transistor degradation model for OBD, based on empirical data from researchers at IBM. This section also presents an HSPICE characterization of the effects of OBD on gates in a 90 nm standard cell library from a major technology vendor.

2.1 Gate Oxide Breakdown

OBD, also known as time dependent dielectric breakdown (TDDB), is caused by the formation of a conductive path through the gate oxide of a CMOS transistor. The progression of OBD causes an increasing leakage current through the gate oxide of devices that eventually leads to oxide failure, rendering the device unresponsive to input stimuli [36, 21, 20]. Sune and Wu showed that there is a significant amount of time required for the OBD leakage current to reach a level capable of affecting circuit performance [36]. This suggests that there is a window of opportunity to detect the onset of OBD before oxide leakage levels compromise the operation of devices and cause timing failures.

The modeling of OBD conducted in this work is based upon the experimental results of Rodriguez et al. [26]. The change in gate oxide current resulting from OBD is modeled by the power-law expression in Equation 1:

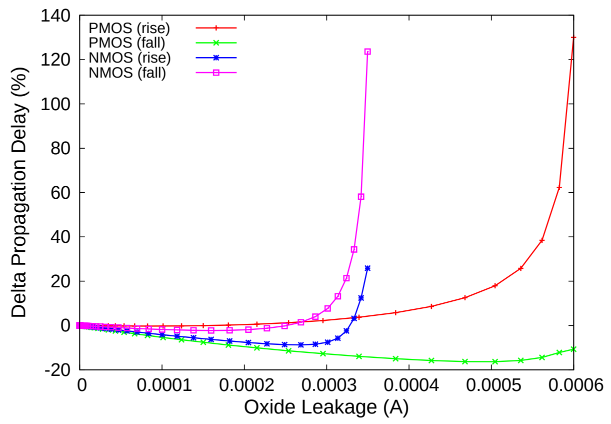
$$\Delta I_{gate} = K(V_{gd})^p \tag{1}$$

The change in gate-to-drain (gate-to-source) current is represented as a function of a linear scaling factor K, the gate-to-drain (gate-to-source) voltage V_{gd} (V_{gs}), and a power-law exponent, p. Both fitting parameters K and p vary depending on the amount of degradation experienced by the transistor in question. However, for much of the empirical data collected in [26], selecting a $p = 5.0$, while still allowing K to track the degree of degradation, resulted in a consistent fit. This is the model for device degradation used in this work.

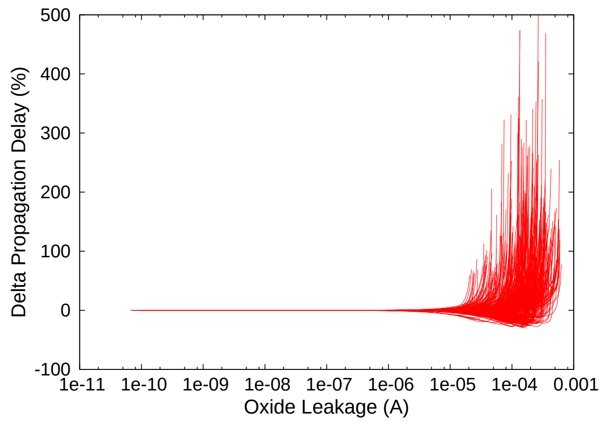
2.2 HSPICE Analysis

To facilitate modeling the effects of OBD-induced degradation in HSPICE, the BSIM4 gate leakage model [11] for gate-to-drain and gate-to-source oxide leakage is modified to accommodate the scaling factor from Equation 1. Using this leakage model, an HSPICE testbench was created to simulate the effects of OBD on propagation delay within logic circuits. The testbench consists of an ideal voltage source driving an undegraded copy of the gate under test, which drives the gate under test, which drives another undegraded copy of the gate under test. This testbench allows the simulations to capture both the loading effects a degraded device presents to nodes on the upstream path, as well as the ability of downstream nodes to regenerate a degraded signal.

For each type of logic gate within the cell library, one transistor at a time is selected from the gate and its leakage model is replaced with the modified BSIM4 model. For each transistor that is being degraded, all input to output transitions are simulated so that for

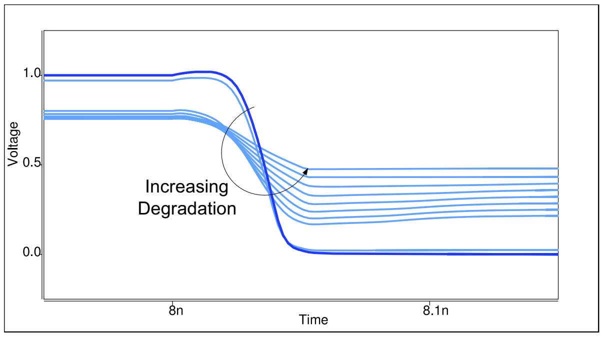


(a) Inverter delay profile.

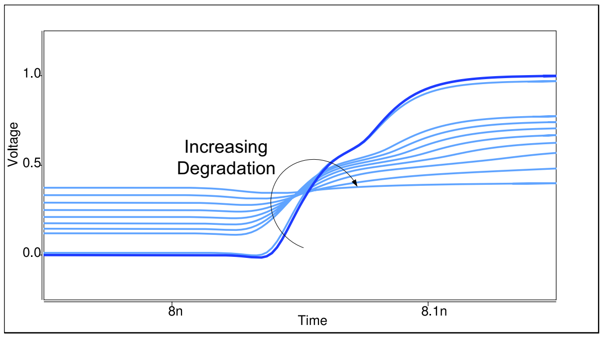


(b) Net effect (rise+fall transitions) on propagation delay for 90 nm standard cells.

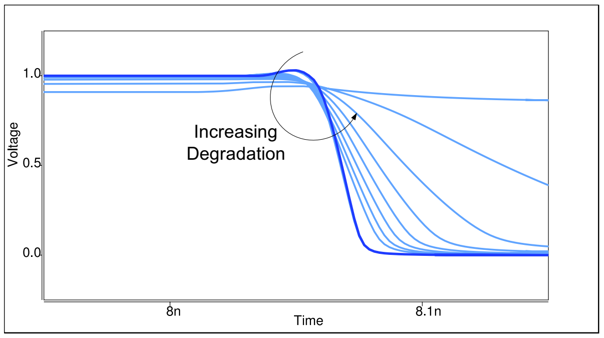
Figure 1: Impact of OBD-induced oxide leakage current on standard cell propagation delays.



(a) Input to inverter cell (t_{cell}).



(b) Output of inverter cell (t_{cell}).



(c) Restored "circuit" output ($t_{circuit}$).

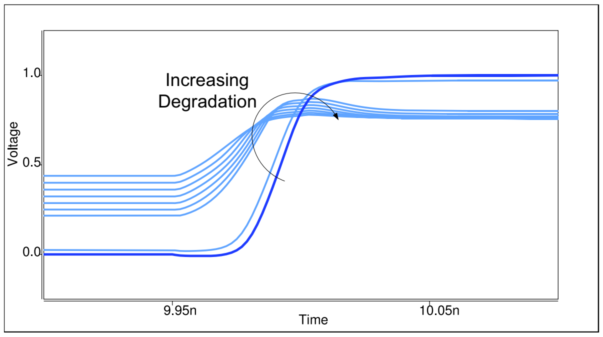
Figure 2: HSPICE simulation traces for inverter with degraded PMOS (slowdown).

every gate characterized, propagation delays corresponding to all possible combinations of degraded transistor, input to output path, and initial input states are captured. For each simulation run, the amount of degradation experienced by the degraded transistor (as modeled by the oxide leakage) is slowly increased until the gate ceases to function (outputs no longer switch).

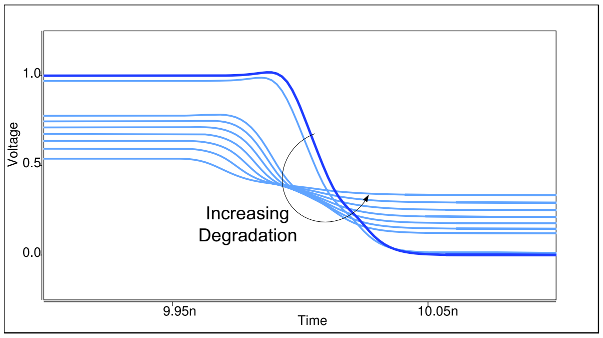
The results of the timing characterization are shown in Figure 1. Figure 1a shows the changes in propagation delay for an average size inverter. The plot highlights the different effects that OBD has on propagation delay depending on the transition direction and location/type of the degraded transistor. Note that for the case when the PMOS (the analogous story is true for the NMOS) is degraded, rising transitions expressed increases in delay while falling transitions showed decreases in delay. A detailed discussion of this phenomenon follows in the next paragraph. Although there are complex dependence relationships affecting the performance impact on rise and fall propagation delays, as a simplifying assumption, the net effect is used in this work. Figure 1b presents the net effect (rising transition + falling transition) of OBD on gates within the cell library. For a given gate a separate curve is shown for each of its transistors. Note that the "net" change in propagation delay is categorically increasing near the end of life for gates within this cell library, irrespective of which internal transistor is degraded.

An examination of Figure 1a reveals that in the case where the PMOS experiences OBD, the rising transition expresses more than a doubling of its nominal delay before the inverter fails to transition. The primary source of this increase in delay is the interaction with the previous stage, a non-degraded inverter, which is subjected to driving the leaky PMOS oxide. Figures 2 and 3 show the voltages at the nodes of interest during the rising and falling transitions of the degraded inverter. The bold traces show the voltage transitions under nominal conditions while the lighter curves are the result of increasing amounts of wearout.

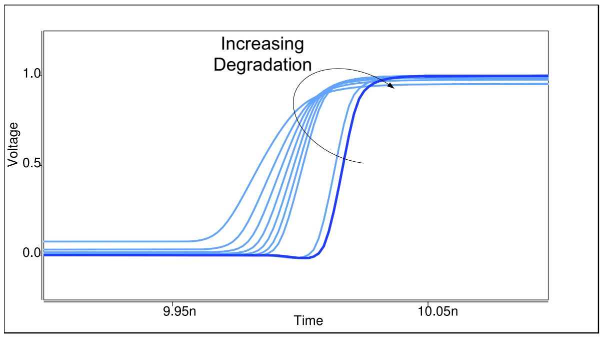
When the input to the inverter under test begins to fall (Figure 2a), the gate-to-source oxide leakage, I_{gs}, through the PMOS device provides additional current to the input node, prolonging the discharge time of the gate through the NMOS of the preceding stage. The gate-to-drain oxide leakage, I_{gd}, initially aids the rising transition, helping to charge up the inverter output. However, as the transition continues and the output begins to rise, this I_{gd} leakage also provides an additional current to the gate node. As with the I_{gs} current, this too increases the time required to drain the remaining charge on the gate. Note also that with large amounts of degradation the input voltage range compresses due to I_{gs} and I_{gd} oxide



(a) Input to inverter cell (t_{cell}).



(b) Output of inverter cell (t_{cell}).



(c) Restored "circuit" output ($t_{circuit}$).

Figure 3: HSPICE simulation traces for inverter with degraded PMOS (speedup).

leakage. Unable to switch from rail-to-rail, the on-currents sustainable by the PMOS and NMOS are significantly reduced, which ultimately contributes to the increase in overall propagation delay (Figure 2c).

Perhaps more surprising is the behavior of the propagation delay during a falling transition of the inverter output (Figure 3). With increasing oxide degradation, the delay of the inverter actually decreases until just prior to functional failure. This behavior is caused by the I_{gs} and I_{gd} leakage currents that help in charging their own gate node, resulting in an earlier rising transition on the input. As a result, despite the degraded on currents due to the compressed gate voltage swing, because the inverter actually "sees" the input transitioning sooner, the net effect is a decrease in the overall propagation delay of the inverter itself (t_{cell}) and ultimately the circuit ($t_{circuit}$).

In summary, at moderate values of oxide degradation, the input voltage on the gate node swings roughly rail-to-rail, allowing normal operation of the inverter. However, during the final stages of oxide OBD, the input voltage range compresses due to I_{gs} and I_{gd} leakage (Figures 3a and 2a), and the current conducted by the PMOS and NMOS devices in the inverter are significantly altered. The significantly reduced output range eventually results in functional failure when the device is no longer capable of driving subsequent stages. Note however, that prior to circuit failure, the stage immediately following the inverter under test is able to completely restore the signal to a full rail swing (Figures 2c and 3c), irrespective of the switching direction.

3. MICROARCHITECTURE-LEVEL WEAROUT ANALYSIS

This section describes how the transistor-level models from the previous section are used to simulate the effects of OBD over time on an embedded microprocessor core. The section begins by describing the processor core studied in this work along with the synthesis flow used in its implementation and thermal analysis. This is followed by a discussion of MTTF calculations and a description of the approach used to conduct Monte Carlo simulations of the effects of OBD. A discussion of OBD's impact on propagation delay at the microarchitectural level concludes the section.

3.1 Microprocessor Implementation

The testbed used to conduct wearout experiments was a Verilog model of the OpenRISC 1200 (OR1200) CPU core [3]. The OR1200 is an open-source, embedded-style, 32-bit, Harvard architecture that implements the ORBIS32 instruction set. The microprocessor contains a single-issue, 5-stage pipeline, with direct mapped 8KB instruction and data caches, and virtual memory support. This microprocessor core has been used in a number of commercial products and is capable of running the μClinux operating system.

The OR1200 core was synthesized using Synopsys Design Compiler, with a cell library characterized for a 90 nm process and a clock period of 2.5 ns (400 MHz). Cadence First Encounter was used to conduct floorplanning, cell placement, clock tree synthesis, and routing. This design flow provided accurate timing information (cell and interconnect delays) and circuit parasitics (resistance and capacitance values) for the entire OR1200 core. The floorplan along with details of the implementation is shown in Figure 4. Note that although the OR1200 microprocessor core shown in Figure 4 is a relatively small design, it's area and power requirements are comparable to that of an ARM9 microprocessor. The final synthesis of the OR1200 appropriates a timing guard band of 250 ps (10% of the clock cycle time) to mimic a commodity processor and to ensure that the wearout simulations do not prematurely cause timing violations.

3.2 Power, Temperature, and MTTF Calculations

The MTTF due to OBD is dependent on many factors, the most significant being oxide thickness, operating voltage, and temperature. To quantify the MTTF of devices undergoing OBD, this work uses the empirical model described in [33], which is based on experimental data collected at IBM [39]. This model is presented in Equation 2.

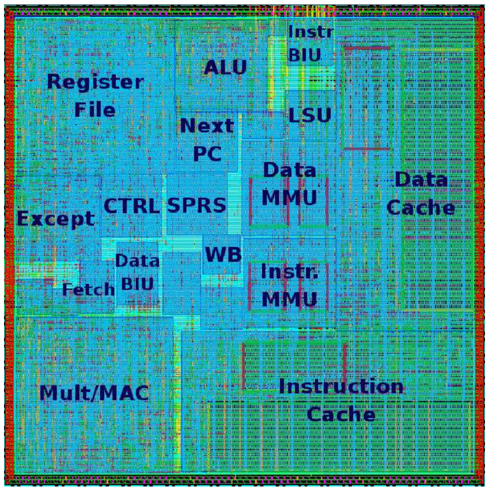
$$MTTF_{OBD} \propto (\frac{1}{V})^{(a-bT)} e^{\frac{(X+\frac{Y}{T}+ZT)}{kT}} \quad (2)$$

where,

- V = operating voltage
- T = temperature

OR1200 Core	
Area	1.0 mm^2
Power	123.9 mW
Clock Frequency	400 MHz
Data Cache Size	8 KB
Instruction Cache Size	8 KB
Logic Cells	24,000
Technology Node	90 nm
Operating Voltage	1.0 V

(a) Implementation details for the OR1200 microprocessor.



(b) Overlay of the OR1200 floorplan on top of the placed and routed implementation of the CPU core.

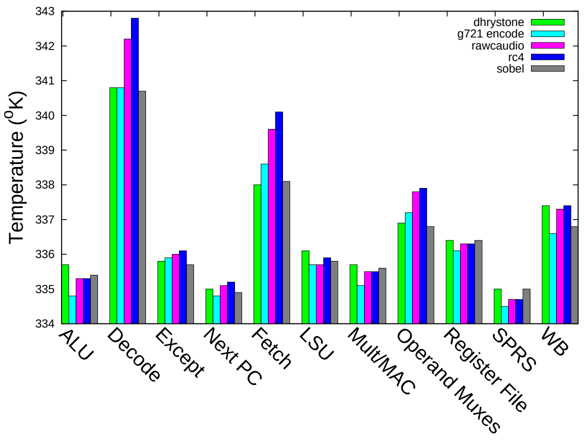
Figure 4: OpenRisc1200 embedded microprocessor.

- k = Boltzmann's constant
- a, b, X, Y, and Z are all fitting parameters based on [33]

In order to calculate the MTTF for devices within the OR1200 core, gate-level activity data was generated by simulating the execution of a set of benchmarks[1] running on a synthesized netlist using Synopsys VCS. This activity information, along with the parasitic data generated during placement and routing, was then used by Synopsys PrimePower to generate a per-benchmark power trace. The power trace and floorplan were in turn processed by HotSpot [29], a block level temperature analysis tool, to produce a dynamic temperature trace and a steady state temperature for each structure within the design.

Once the activity and temperature data were derived, the MTTF for each logic gate in the design was calculated using Equation 2 with the temperature and activity data for each benchmark. A per-module MTTF is calculated by identifying the minimum MTTF across all logic gates within each top-level module of the OR1200 core. These per-module MTTF values are later used to parametrize the statistical distribution of failures used in Monte Carlo simulations of OBD effects. Figure 5 presents the steady state temperatures and MTTF values of different structures within the CPU core for the five benchmarks.

[1]Five benchmarks were studied to represent a range of computational behavior for embedded systems: dhrystone - a synthetic integer benchmark; g721encode and rawcaudio from the MediaBench suite; rc4 - an encryption algorithm; and sobel - an image edge detection algorithm.



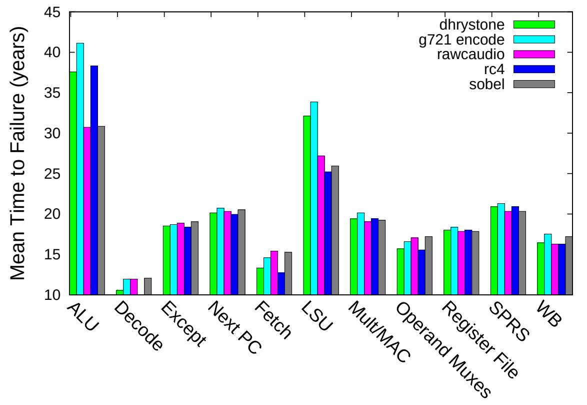


Figure 5: Derived workload-dependent steady state temperature and MTTF for the OR1200 CPU core. An ambient temperature of 333K was used for Hotspot.

Figure 5 highlights the correlation between MTTF and temperature. Structures with the highest temperatures tended to have the smallest MTTFs, meaning that they were most likely to wearout first. For example, the decode unit, with a maximum temperature about 3°K higher than any other structure on the chip, would likely be the first structure to fail. Somewhat surprisingly, the ALU had a relatively low temperature, resulting in a long MTTF. Upon further investigation, it was found that across most benchmark executions, less than 50% of dynamic instructions exercised the ALU, and furthermore, about 20% of the instructions that actually required the ALU were simple logic operations and not computationally intensive additions or subtractions. These circumstances led to a relatively low utilization and ultimately lower temperatures. It is important to note that although this work focuses on a simplified CPU model, the proposed wearout detection technique is not coupled to a particular microprocessor design or implementation, but rather relies upon the general circuit-level trends suggested by the HSPICE simulations. In fact, a more aggressive, high performance microprocessor is likely to have more dramatic hotspots, which would only serve to exaggerate the trends that motivate the WDU design presented in this work.

3.3 Wearout Simulation

As demonstrated in Section 2, progressive wearout phenomena

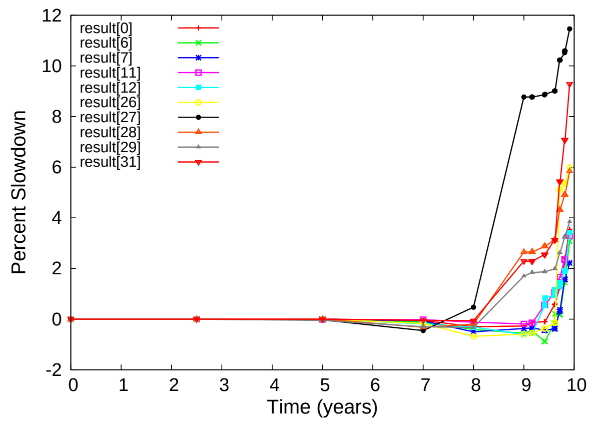


Figure 6: The observed slowdown of signals from the ALU result bus as a result of OBD effects over the lifetime of one instance of an OR1200 processor core.

(OBD in particular) have a significant impact on circuit-level timing. Work done by Linder and Stathis [20] has shown that OBD-induced gate leakage obeys an exponential growth rate with age:

$$\Delta I_{OBD}(t) = I_{OBD_0} \cdot e^{t/\gamma} \quad (3)$$

where,

- $I_{OBD}(t)$: the amount of leakage current at time t
- I_{OBD_0}: the initial amount of leakage current at time 0
- γ: varied to model a "fast" or "slow" exponential growth

Monte Carlo simulations of the OBD effects on a distribution of microprocessors in the field are conducted by coupling the leakage model, shown in Equation 3, with the model for MTTF from Equation 2. For every logic gate within each microprocessor simulated, the time when the first initial breakdown path is formed in the oxide, τ_{BD}, is calculated using a Weibull distribution with α equal to the gate's MTTF and $\beta = 1.0$, consistent with [20]. The growth rate γ is then taken from a uniform distribution of $+/- 10\%$ of τ_{BD}, consistent with a slow growth rate, as in [20].

By integrating the model for OBD failure times and this leakage growth model, a statistically accurate picture of the effects of OBD-induced leakage for every gate within the OR1200 core (across a population of chips) is derived. This new model is then used to generate age-dependent performance data for each gate within the population of processors in the Monte Carlo simulations. The performance information is then annotated onto the synthesized netlist and custom signal monitoring handlers are used to measure the signal propagation delays at the output of various modules within the design. The process of annotation and monitoring is repeated for every processor in the population at regular time intervals over the simulated lifetime of each processor.

To demonstrate how OBD can affect the timing of microarchitectural structures, Figure 6 shows the results of one sample of an OR1200 core from the Monte Carlo simulations. This figure shows the amount of performance degradation observed at the output of the ALU for a subset of signals from the result bus. This figure illustrates the general trend of slowdown across output signals from microarchitectural structures. The following section discusses how this trend is leveraged to conduct wearout detection and failure prediction.

4. WEAROUT DETECTION

In this section, the delay trends for microarchitectural structures observed in Section 3 are leveraged to propose a novel technique for predicting wearout-induced failures. The technique consists of two logical steps: online delay monitoring and statistical analysis of delay data. In the following subsection, a circuit for conducting online delay sampling is presented. Next, the algorithm used for statistical analysis, TRIX, is presented, and its applicability to wearout detection is discussed. Finally, two potential implementations for the statistical analysis of delay profiles are proposed, one in hardware and the other in software.

4.1 Online Delay Profiling

In this section, a self-calibrating circuit for online delay profiling is presented. A schematic diagram of the online delay profiling unit (DPU) is shown in Figure 7. The DPU is used to measure the time that elapses after a circuit's output signal stabilizes until the next positive clock edge (slack time). It is important to note that even for critical paths within the design, some slack time exists because of guard bands provisioned into the design for worst-case environmental variation and signal degradation due to wearout. The DPU design consists of three distinct stages. The first stage of the DPU is an arbiter that determines which one of the (potentially many) input signals to the DPU will be profiled. The second stage of the DPU generates an approximation of the available slack time. The final stage of the DPU is an accumulator that totals a sample of 4096 signal transition latency measurements, and uses this measurement as a point estimate for the amount of available slack in the circuit for the given input signal.

The first stage fulfills the simple purpose of enabling the DPU to monitor delay information for multiple output signals from a given structure. This stage is a simple arbiter that determines which signal will be monitored. The area of this structure scales linearly (though very slowly) with the number of output signals being monitored. The effects of scaling on area and power are discussed later in Section 5.

The purpose of the second stage of the DPU is to obtain a coarse-grained profile of the amount of slack at the end of a given clock period. The signal being monitored by the DPU is connected to a series of delay buffers. Each delay buffer in this series feeds one bit in a vector of registers (labeled 'A' in Figure 7) such that the signal arrival time at each register in this vector is monotonically increasing. At the positive edge of the clock, some of these registers will capture the correct value of the module output, while others will store an incorrect value (the previous signal value). This situation arises because the propagation delay imposed by the sequence of delay buffers causes the output signal to arrive after the latching window for a subset of these registers. The value stored at each of the registers is then compared with a copy of the correct output value, which is stored in the register labeled 'B'. The XOR of each delayed register value with the correct value produces a bit vector that represents the propagation delay of the path exercised for that particular cycle. In addition, the output signal value from the previous cycle is stored in the register labeled 'C', and is used to identify cycles during which the module output actually experiences a transition. This ensures that cycles during which the output is idle do not bias the latency sample. As a module's performance degrades due to wearout, the signal latency seen at its outputs increases, fewer comparisons will succeed, and the value reported at the output of the vector of XOR gates will increase.

In the third stage of the DPU, a point estimate of the mean propagation latency for a given output signal is calculated by accumulating 4096 signal arrival times. The accumulation of 4096 arrival

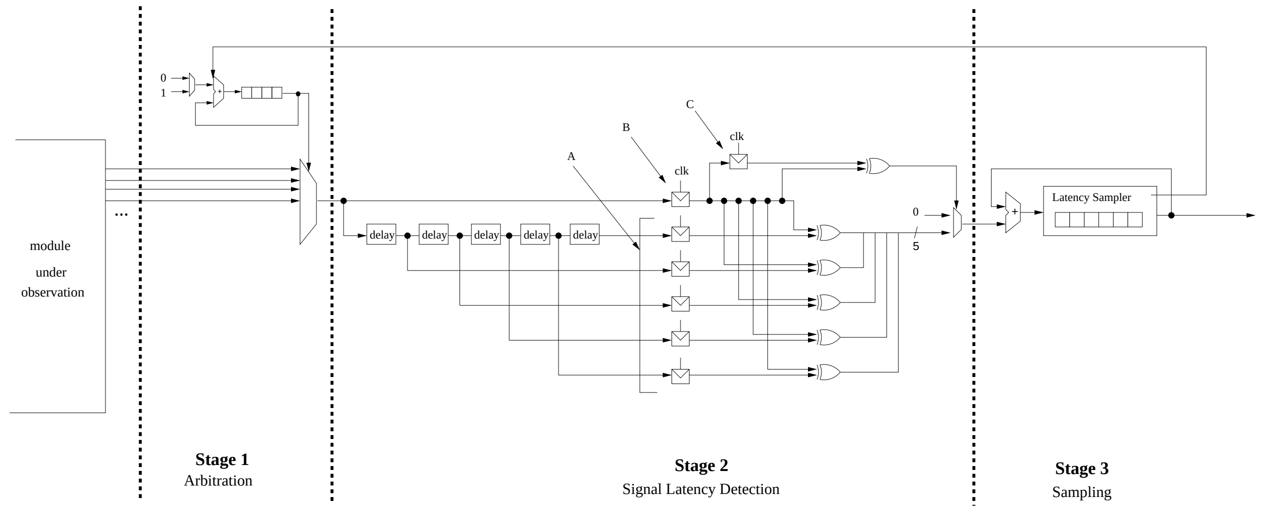


Figure 7: Online delay profiling unit.

times is used to smooth out the variation in path delays that are caused by variation in the module input, and the sample size 4096 is used because it is a power of two and allows for efficient division by shifting.

There are multiple ways in which this sampled mean propagation latency may be utilized by a system for failure prediction. In the next subsection, an algorithm is presented for this purpose that may be implemented either in specialized hardware or software.

4.2 Failure Prediction Algorithm

In order to capitalize on the trend of divergence between the signal propagation latency observed during the early stages of the microprocessor's lifetime and those observed at the end of life, TRIX (triple-smoothed exponential moving average) [35] analysis is used. TRIX, is a trend analysis technique used to measure momentum in financial markets and relies on the composition of three calculations of an exponential moving average (EMA) [10]. The EMA is calculated by combining the current sample value with a fraction of the previous EMA, causing the weight of older sample values to decay exponentially over time. The calculation of EMA is given as:

$$EMA = \alpha \times sample + (1 - \alpha)EMA_{prev} \tag{4}$$

The use of TRIX, rather than the EMA, provides two significant benefits. First, TRIX provides an excellent filter of noise within the data stream because the composed applications of the EMA smooth out aberrant data points that may be caused by dynamic variation, such as temperature or power fluctuations (discussed in Section 5.2). Second, the TRIX value tends to provide a better leading indicator of sample trends. The equations for computing the TRIX value are:

$$\begin{aligned} EMA_1 &= \alpha(sample - EMA_{1prev}) + EMA_{1prev} \\ EMA_2 &= \alpha(EMA_1 - EMA_{2prev}) + EMA_{2prev} \\ TRIX &= \alpha(EMA_2 - TRIX_{prev}) + TRIX_{prev} \end{aligned} \tag{5}$$

TRIX calculation is recursive and parametrized by the weight, α, which dictates the amount of emphasis placed on older sample values. Figure 8a demonstrates the impact of different α values on the amount of weight given to historical samples. This figure demonstrates that small α values tend to favor older samples, while larger α values reflect local trends. The wearout detection algorithm presented in this work relies on the calculation of two TRIX values using different α's to identify when the local trends in the observed signal latency begin to diverge from the historical trends (biased toward early-life timing). Figure 8b shows the effect of different α values on the TRIX analysis of ALU result bit 0. Figure 8b presents the TRIX calculations for six different α values as well as the long-term running average and local point average of signal over the lifetime of the microprocessor. This data demonstrates that TRIX calculation using $\alpha = 1/2^{10}$ provides an accurate estimate of the running average (of latencies for a signal) over the lifetime of the chip, and does so without the overhead of maintaining a large history. Further, this figure shows that a TRIX calculation with $\alpha = 0.8$ provides a good indicator of the local sample latency at a given point in the microprocessor's lifetime.

The next subsection describes two potential implementations that bring together the DPU and this statistical analysis technique in order to predict the failure of structures within a processor core.

4.3 Implementation Details

In order to accurately detect the progression of wearout and predict when structures are likely to fail, this work proposes the use of the DPU in conjunction with TRIX analysis. In the following subsections, two techniques for building systems with wearout prediction mechanisms are proposed. The first technique is a hardware-only approach, where both online delay profiling and TRIX analysis are conducted together in a specialized hardware unit called the wearout detection unit (WDU). The second technique is a hybrid approach requiring fewer resources where delay profiling is conducted in hardware, but TRIX analysis is conducted in software, either in the operating system or in firmware. In Section 5, we discuss the hardware costs in terms of area and power for each of these implementations, as well how the WDU scales as it is used to monitor an increasing number of signals.

4.3.1 *Hardware-only Implementation*

The design of the WDU is presented in Figure 9 and consists of three distinct stages. The first stage is comprised of the delay profiling unit described in Section 4.1, while the second stage is responsible for conducting the TRIX analysis discussed in Sec-

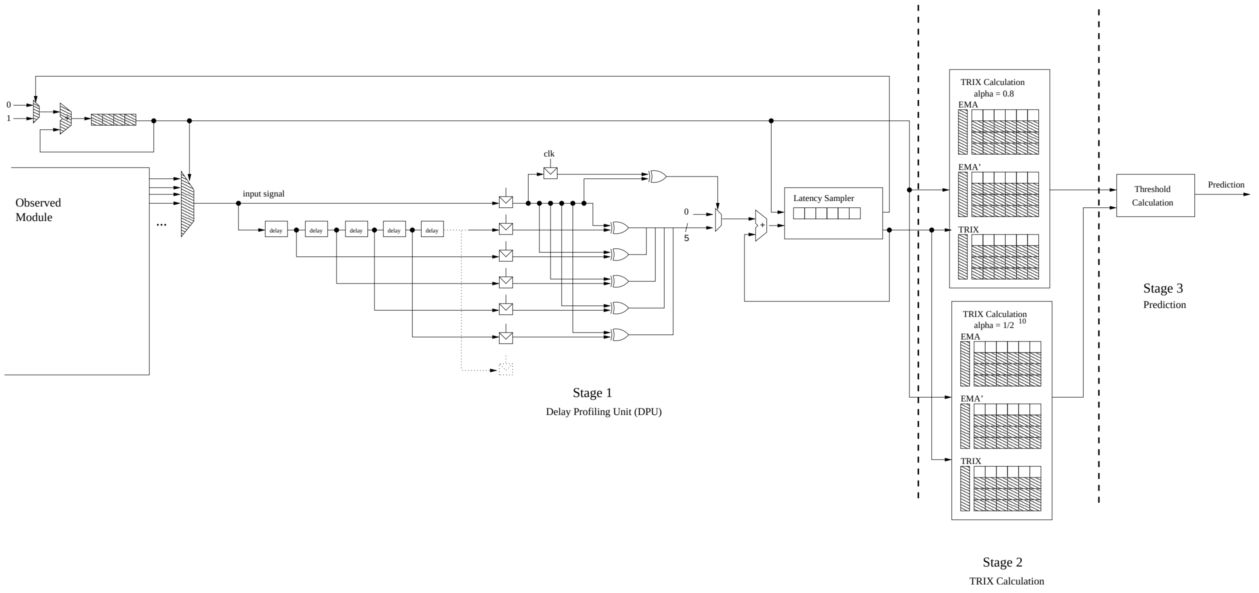


Figure 9: Design and organization of the wearout detection unit.

tion 4.2, and the third stage conducts threshold analysis to identify significant divergences in latency trends. The shaded structures in this diagram represent those components that would scale with the number of signals being monitored. The remainder of this section discusses the implementation details of stage two and three of this design, and the required resources for their implementation.

In stage two of the WDU, two TRIX values are computed: a locally-biased value, $TRIX_l$, and a historically-biased value, $TRIX_g$. These are calculated using α values of 0.8 and $1/2^{10}$, respectively. It is important to note that the value of α is dependent on the sample rate and sample period. In this work, we assume a sample rate of three to five samples per day over an expected 10 year lifetime. Also, the long incubation periods for many of the common wearout mechanisms require the computed TRIX values to routinely be saved into a small area of nonvolatile storage, such as flash memory.

Since the TRIX consists of three identical EMA calculations, the impact of Stage 2 on both area and power can be minimized by spanning the calculation of the TRIX values over multiple cycles and only synthesizing a single instance of the EMA calculation hardware. Section 5 describes the area and power overhead for the WDU in more detail.

The third stage of the WDU receives $TRIX_l$ and $TRIX_g$ values from the previous stage and is responsible for predicting a failure if the difference between these two values exceeds a given threshold. The simulations conducted in this work indicate that a 5% difference between $TRIX_l$ and $TRIX_g$ is almost universally indicative of a structure nearing failure. It is envisioned that this prediction would be used to enable a cold spare device, or notify a higher-level configuration manager of a potentially failing structure within the core. An analysis of the accuracy of this threshold prediction is presented in Section 5.

4.3.2 *Hardware/Software Hybrid Implementation*

In order to alleviate some of the scaling problems and resource requirements of a hardware only technique, a hardware/software hybrid technique can be used. In this system, the DPU is still implemented in hardware, while the TRIX analysis is performed in software by the operating system or system firmware. In this configuration, a set of dedicated registers for maintaining the latency samples for different modules within the design are used. These dedicated registers are similar to the performance counters used in modern day processors. The system software then regularly samples these counters and can store the calculated TRIX values to disk or other non-volatile storage.

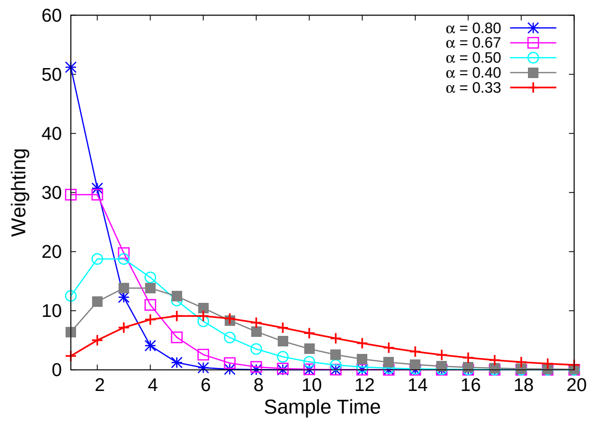
This hardware/software hybrid design has multiple benefits over the hardware-only approach. In the hardware-only approach, the TRIX calculation, as well as the α parametrization values are hardwired into the design, meaning that across different technology generations with different wearout progression rates, different WDU implementations will be necessary. However, in the hybrid approach, the TRIX parametrization is easily modified for use in a variety of systems. Another benefit is that the hybrid implementation consumes less power and has a smaller area footprint with better scaling properties than the hardware-only design.

5. EXPERIMENTAL ANALYSIS

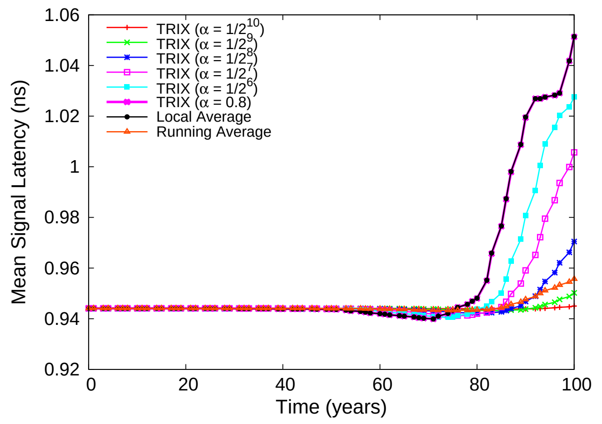
This section provides a detailed analysis of the proposed WDU for both the hardware-only and hybrid implementations, the area and power overhead for implementation, and its efficacy in predicting failure.

5.1 Overhead and Accuracy

Figure 10 demonstrates the area and power requirements for a WDU and a DPU (for the hybrid approach) implemented in Verilog and synthesized using a 90 nm standard cell library, designed to monitor multiple output signals for a structure. The x-axis represents the number of signals being monitored and the y-axis represents the overhead in terms of area or power. Figure 10a demonstrates that the WDU scales poorly in terms of area, and Figure 10b shows analagous results for power. This behavior is largely because the amount of storage within the WDU increases linearly with the number of signals being monitored. In contrast, the DPU scales well in both area and power with an increasing number of signals being monitored because only the logic for the arbiter scales with an increasing number of signals, and this increase in logic is for



(a) Impact of α value on the weighting of old sample values.



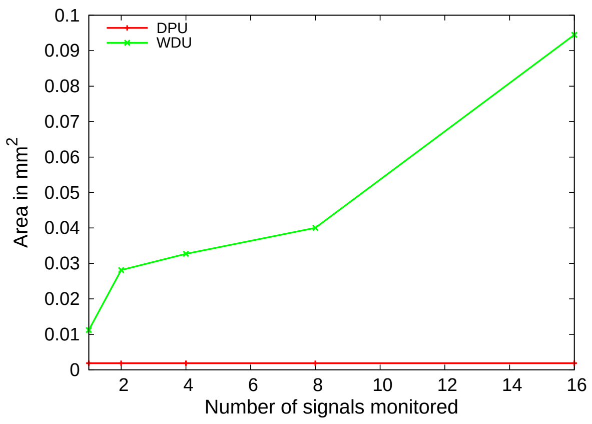
(b) Impact of α value on the tracking of a signal undergoing OBD degradation effects.

Figure 8: Sensitivity analysis of TRIX parametrization.

the most part negligible. This implies that the hybrid prediction technique can be implemented at a much lower design cost.

In order to evaluate the efficacy of TRIX analysis in predicting failure, a large number of Monte Carlo wearout simulations were conducted using the Weibull distribution and failure model presented in Section 3.2. Figure 11a demonstrates the relative time at which failure was predicted for a variety of structures within the processor core for the population of microprocessors used in this Monte Carlo simulation. The error bars in this figure represent the standard deviation of these values. Across all simulations, failure was predicted within 20% of the time of failure for the device. This typically amounted to slightly less than two years of remaining life before the device ultimately failed. Two extreme cases were the Next PC module and the LSU, where the failure prediction was often almost too optimistic, with many of the failure predictions being made with only about 1% or about 4 days of the structure's life remaining. On the opposite end of the spectrum, failure of the register file was often predicted with more than 15% of the lifetime remaining, meaning that some usable life would be wasted in a cold-sparing situation.

Figure 11b demonstrates the percentage of signals that caused predictions to be raised for each module before the module failed. In general, the percentage of outputs flagged at the time of failure varied widely. This can be attributed to a number of factors. First, the Weibull distribution used to model the time of first breakdown for each gate within the design has a moderate amount of variance,



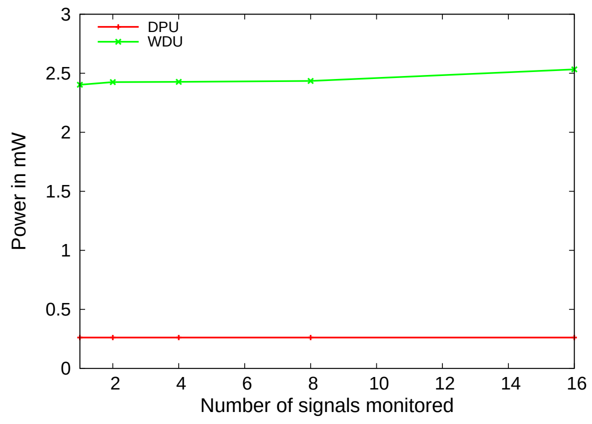


Figure 10: Scaling of the WDU and DPU area and power as the number of signals monitored scales.

as does the uniform distribution used to model the growth rate of leakage from the time of first breakdown. Also, because some gates experience speedup in the early stages of wearout before they ultimately begin to slow down, there are competing effects between gates at different stages of wearout early in the breakdown period.

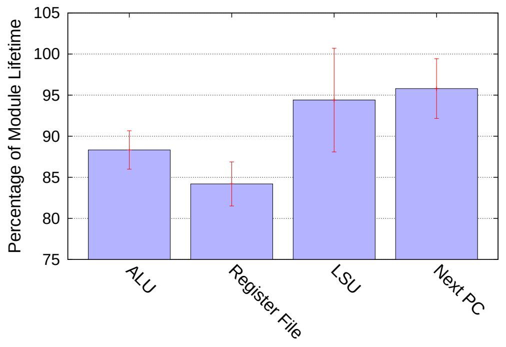
5.2 Dynamic Variations

Dynamic environmental variations, such as temperature spikes, power surges, and clock jitter, can each have an impact on circuit-level timing, potentially affecting the operation of the WDU. Here, we briefly discuss some of the sources of dynamic variation and their impact on the WDU's efficacy.

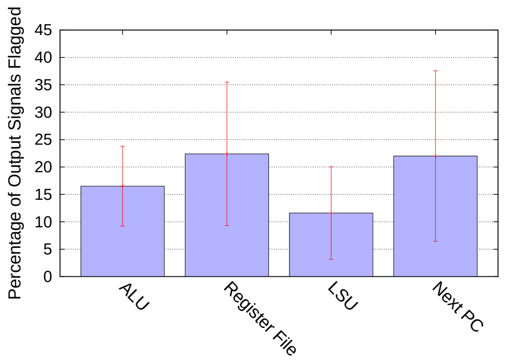
Temperature is a well known factor in calculating device delay, where higher temperatures typically increase the response time for logic cells. Figure 12 demonstrates the increase in response time for a selection of logic gates[2] over a wide range of temperatures. This figure shows that over an interval of 50^{o}C, the increase in response time amounts to approximately 3.4%.

Another source of variation is clock jitter. In general, there are three types of jitter: absolute jitter, period jitter, and cycle-to-cycle jitter. Of these, cycle-to-cycle jitter is the only form of jitter that may potentially affect the WDU. Cycle-to-cycle jitter is defined as the difference in length between any two adjacent clock periods and may be both positive (cycle 2 longer than cycle 1) or negative (cycle 2 shorter than cycle 1). Statistically, jitter measurements exhibit a random distribution with a mean value approaching zero [40].

[2]The gate models were taken from the 90 nm library and simulated using HSPICE.



(a) Average percentage of the lifetime at which failure predictions are made with error bars representing the standard deviation of the population.



(b) Percentage of the output signals that were flagged at the time of failure for each module with error bars representing the standard deviation of the population.

Figure 11: Analysis of TRIX analysis efficacy in predicting failure.

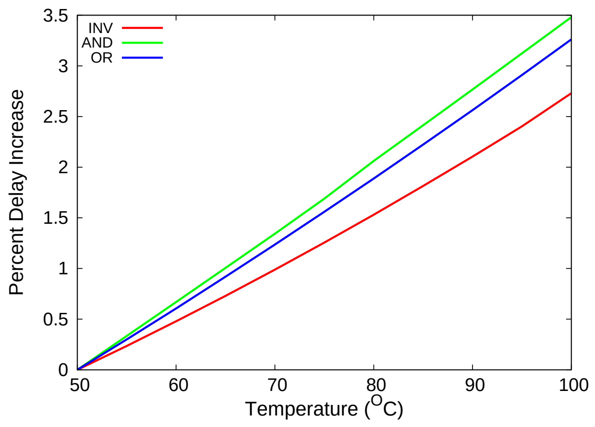


Figure 12: Impact of temperature on logic gate delay.

In general, the sampling techniques employed for failure prediction are sufficient to smooth out the effects of dynamic variation described. For example, a conservative, linear scaling of temperature effects on the single inverter delay to a 3.4% increase in module output delay does not present a sufficient magnitude of variance to overcome the 5% threshold required for the WDU to predict failure. Also, because the expected variation due to both clock jitter and temperature will exhibit a mean value of zero (i.e., temperature is expected to fluctuate both above and below the mean value), statistical sampling of latency values should minimize the impact of these variations. To further this point, since the TRIX calculation acts as a three-phase low-pass filter, the worst case dynamic variations would need to cause latency samples to exceed the stored $TRIX_g$ value by greater than 5% over the course of more than 12 successive sample periods, corresponding to over four days of operation.

6. RELATED WORK

Issues in technology scaling and process variation have raised concerns for reliability in future microprocessor generations. Recent research work has attempted to diagnose and, in some cases, reconfigure the processing core to increase operational lifetime. In this section, we briefly discuss this related work and how it has influenced the direction of this paper.

As mentioned in Section 1, much of the research into failure detection relies upon redundancy, either in time or space. One such example of hardware redundancy is DIVA [8], which targets soft error detection and online correction. It strives to provide a low cost alternative to the full scale replication employed by traditional techniques like triple-modular redundancy. The system utilizes a simple in-order core to monitor the execution from a large high performance superscalar processor. The smaller checker core recomputes instructions before they commit and initiates a pipeline flush within the main processor whenever it detects an incorrect computation. Although this technique proves useful in certain contexts, the second microprocessor requires significant design/verification effort to build and incurs additional area overhead.

Bower et al. [15] extends the DIVA work by presenting a method for detecting and diagnosing hard failures using a DIVA checker. The proposed technique relies on maintaining counters for major architectural structures in the main microprocessor and associating every instance of incorrect execution detected by the DIVA checker to a particular structure. When the number of faults attributed to a particular unit exceeds a predefined threshold, it is deemed faulty and decommissioned. The system is then reconfigured, and in the presence of cold spares, can extend the useful life of the processor. Related work by Shivakumar et al. [28] argues that even without additional spares the existing redundancy within modern processors can be exploited to tolerate defects and increase yield through reconfiguration.

Research by Vijaykumar [18, 37] and Falsafi [22, 30] attempt to exploit the redundant, and often idle, resources of a superscalar processor to enhance reliability by utilizing these extra units to verify computations during periods of low resource demand. This technique represents an example of the time redundant computation alluded to in Section 1. It leverages work by the Slipstream group [27, 23] on simultaneous redundant multithreading as well as earlier work on instruction reuse [31]. ReStore [38] is another variation on this theme which couples time redundancy with symptom detection to manage the adverse effects of redundant computation by triggering replication only when the probability of an error is high.

Srinivasan et al. have also been very active in promoting the need for robust designs that can withstand the wide variety of reliability challenges on the horizon [34]. Their work attempts to accurately model the MTTF of a device over its operating lifetime, facilitating the intelligent application of techniques like dynamic voltage and/or frequency scaling to meet reliability goals. Although some physical models are shared in common, the focus of our paper is not to guarantee that designs can achieve any particular reliability goal, but rather to enable a design to recognize behavior that is symptomatic of wearout induced breakdown allowing it to react accordingly.

Analyzing circuit timing in order to self-tune processor clock frequencies and voltages is also a well studied area. Kehl [19] discusses a technique for re-timing circuits based on the amount of cycle-to-cycle slack existing on worst-case latency paths. The technique presented requires offline testing involving a set of stored test vectors in order to tune the clock frequency. Although the proposed circuit design is similar in nature to the WDU, it only examines the small period of time preceding a clock edge and is only concerned with worst case timing estimation, whereas the WDU employs sampling over a larger time span in order to conduct average case timing analysis. Similarly, Razor [9] is a technique for detecting timing violations using time-delayed redundant latches to determine if operating voltages can be safely lowered. Again, this work studies only worst-case latencies for signals arriving very close to the clock edge.

7. CONCLUSION

In this paper, the device-level effects of oxide breakdown (OBD) on circuit performance are characterized. It is demonstrated that progressive OBD has a non-uniform impact on circuit performance. The results of the circuit-level modeling are then applied to a synthesized implementation of the OR-1200 microprocessor to analyze the effects of OBD at the microarchitectural level. Circuit timing is identified as a common phenomenon that can be tracked to predict the progression of OBD. A self-calibrating circuit for analyzing circuit path delay along with an algorithm for predicting failure using this data is proposed. Results show that our failure prediction algorithm is successful in identifying wearout and flagging outputs that suffer a trend of increasing delay over time.

8. ACKNOWLEDGMENTS

We thank the anonymous referees for their comments and suggestions. We would also like to thank David Blaauw and Eric Karl for their insightful discussions and feedback. This research was supported by ARM Limited, the National Science Foundation ITR grant CCR-0325898, and the Gigascale Systems Research Center, one of five research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program.

9. REFERENCES

[1] Gartner data systems conference, Dec. 2005.

[2] International technology roadmap for semiconductors, 2005.

[3] Openrisc 1200, 2006. http://www.opencores.org/projects.cgi/web/or1k/openrisc_1200.

[4] Ridgetop group, 2006. http://www.ridgetop-group.com/.

[5] M. Agarwal, B. Paul, and S. Mitra. Circuit failure prediction and its application to transistor aging. In *Proc. of the 2007 IEEE VLSI Test Symposium*, page To appear, Apr. 2007.

[6] E. R. Alliance. Online survey results: 2001 cost of downtime, 2001.

[7] J. S. S. T. Association. Failure mechanisms and models for semiconductor devices. Technical Report JEP122C, JEDEC Solid State Technology Association, Mar. 2006.

[8] T. Austin. Diva: a reliable substrate for deep submicron microarchitecture design. In *Proc. of the 32nd Annual International Symposium on Microarchitecture*, pages 196–207, 1999.

[9] T. Austin, D. Blaauw, T. Mudge, and K. Flautner. Making typical silicon matter with razor. *IEEE Computer*, 37(3):57–65, Mar. 2004.

[10] M. Batty. Monitoring an exponential smoothing forecasting system. *Operational Research Quaterly*, 20(3):319–325, 1969.

[11] B. G. U. Berkeley. Bsim4 mosfet model, 2007.

[12] D. Bernick, B. Bruckert, P. D. Vigna, D. Garcia, R. Jardine, J. Klecka, and J. Smullen. Nonstop Advanced Architecture. In *International Conference on Dependable Systems and Networks*, pages 12–21, June 2005.

[13] J. A. Blome, S. Feng, S. Gupta, and S. Mahlke. Online timing analysis for wearout detection. In *Proc. of the 2nd Workshop on Architectural Reliability (WAR)*, pages 51–60, 2006.

[14] S. Borkar. Designing reliable systems from unreliable components: The challenges of transistor variability and degradation. *IEEE Micro*, 25(6):10–16, 2005.

[15] F. A. Bower, D. J. Sorin, and S. Ozev. A mechanism for online diagnosis of hard faults in microprocessors. In *Proc. of the 38th Annual International Symposium on Microarchitecture*, pages 197–208, 2005.

[16] J. Carter, S. Ozev, and D. Sorin. Circuit-level modeling for concurrent testing of operational defects due to gate oxide breadkown. In *Proc. of the 2005 Design, Automation and Test in Europe*, pages 300–305, June 2005.

[17] D. Ernst, N. S. Kim, S. Das, S. Pant, T. Pham, R. Rao, C. Ziesler, D. Blaauw, T. Austin, and T. Mudge. Razor: A low-power pipeline based on circuit-level timing speculation. In *Proc. of the 36th Annual International Symposium on Microarchitecture*, pages 7–18, 2003.

[18] M. Gomaa and T. Vijaykumar. Opportunistic transient-fault detection. In *Proc. of the 32nd Annual International Symposium on Computer Architecture*, pages 172–183, June 2005.

[19] T. Kehl. Hardware self-tuning and circuit performance monitoring. In *Proc. of the 1993 International Conference on Computer Design*, pages 188–192, Oct. 1993.

[20] B. P. Linder and J. H. Stathis. Statistics of progressive breakdown in ultra-thin oxides. *Microelectronic Engineering*, 72(1-4):24–28, 2004.

[21] F. Monsieur, E. Vincent, D. Roy, S. Bruyere, J. C. Vildeuil, G. Pananakakis, and G. Ghibaudo. A thorough investigation of progressive breakdown in ultra-thin oxides. physical understanding and application for industrial reliability assessment. In *Proc. of the 2002 International Reliability Physics Symposium*, pages 45–54, Apr. 2002.

[22] J. Ray, J. Hoe, and B. Falsafi. Dual use of superscalar datapath for transient-fault detection and recovery. In *Proc. of the 34th Annual International Symposium on Microarchitecture*, pages 214–224, Dec. 2001.

[23] V. Reddy, S. Parthasarathy, and E. Rotenberg. Understanding prediction-based partial redundant threading for low-overhead, high-coverage fault tolerance. In *14th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 83–94, Oct. 2006.

[24] S. K. Reinhardt and S. S. Mukherjee. Transient fault detection via simulataneous multithreading. In *Proc. of the 27th Annual International Symposium on Computer Architecture*, pages 25–36, June 2000.

[25] G. Reis, J. Chang, N. Vachharajani, R. Rangan, , and D. I.

August. SWIFT: Software implemented fault tolerance. In *Proc. of the 2005 International Symposium on Code Generation and Optimization*, pages 243–254, 2005.

[26] R. Rodriguez, J. H. Stathis, and B. P. Linder. Modeling and experimental verification of the effect of gate oxide breakdown on cmos inverters. In *International Reliability Physics Symposium*, pages 11–16, Apr. 2003.

[27] E. Rotenberg. AR-SMT: A microarchitectural approach to fault tolerance in microprocessors. In *International Symposium on Fault Tolerant Computing*, pages 84–91, 1999.

[28] P. Shivakumar, S. Keckler, C. Moore, and D. Burger. Exploiting microarchitectural redundancy for defect tolerance. In *Proc. of the 2003 International Conference on Computer Design*, page 481, Oct. 2003.

[29] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan. Temperature-aware microarchitecture: Modeling and implementation. *ACM Transactions on Architecture and Code Optimization*, 1(1):94–125, 2004.

[30] J. Smolens, J. Kim, J. Hoe, and B. Falsafi. Efficient resource sharing in concurrent error detecting superscalar microarchitectures. In *Proc. of the 37th Annual International Symposium on Microarchitecture*, pages 256–268, Dec. 2004.

[31] A. Sodani and G. Sohi. Dynamic instruction reuse. In *Proc. of the 25th Annual International Symposium on Computer Architecture*, pages 194–205, June 1998.

[32] L. Spainhower and T. Gregg. IBM S/390 Parallel Enterprise Server G5 Fault Tolerance: A Historical Perspective. *IBM Journal of Research and Development*, 43(6):863–873, 1999.

[33] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. The case for lifetime reliability-aware microprocessors. In *Proc. of the 31st Annual International Symposium on Computer Architecture*, pages 276–287, June 2004.

[34] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. Exploiting structural duplication for lifetime reliability enhancement. In *Proc. of the 32nd Annual International Symposium on Computer Architecture*, pages 520–531, June 2005.

[35] StockCharts.com. *TRIX*, Oct. 2006. http://stockcharts.com/education/IndicatorAnalysis/indic_trix.htm.

[36] J. Sune and E. Wu. From oxide breakdown to device failure: an overview of post-breakdown phenomena in ultrathin gate oxides. In *International Conference on Integrated Circuit Design and Technology*, pages 1–6, May 2006.

[37] T. Vijaykumar, I. Pomeranz, and K. Cheng. Transient-fault recovery via simultaneous multithreading. In *Proc. of the 29th Annual International Symposium on Computer Architecture*, pages 87–98, May 2002.

[38] N. Wang and S. Patel. Restore: Symptom based soft error detection in microprocessors. In *International Conference on Dependable Systems and Networks*, pages 30–39, June 2005.

[39] E. Wu et al. Interplay of voltage and temperature acceleration of oxide breakdown for ultra-thin gate oxides. *Solid-State Electronics*, 46:1787–1798, 2002.

[40] T. J. Yamaguchi, M. Soma, D. Halter, J. Nissen, R. Raina, M. Ishida, and T. Watanabe. Jitter measurements of a powerpc microprocessor using an analytic signal method. In *Proc. of the 2000 International Test Conference*, pages 955–964, 2000.

[41] X. Yang, E. Weglarz, and K. Saluja. On nbti degradation process in digital logic circuits. In *Proc. of the 2007 International Conference on VLSI Design*, pages 723–730, Jan. 2007.