Credit-Based Dynamic Reliability Management Using Online Wearout Detection

John Oliver
Cal Poly State University
San Luis Obispo, CA
jyoliver@calpoly.edu

Venkatesh Akella
University of California
Davis, CA
akella@ece.ucdavis.edu

Rajeevan Amirtharajah
University of California
Davis, CA
ramirtha@ece.ucdavis.edu

Frederic T. Chong
University of California
Santa Barbara, CA
chong@cs.ucdavis.edu

ABSTRACT

As circuit geometries continue to shrink, and supply voltages remain relatively constant, circuit wearout becomes a concern. We propose that the relative reliability of the circuits of a processor be exposed to the operating system, and be managed by a credit-based wearout monitor. This wearout monitor receives dynamic updates of the reliability of circuits through the use of stability detector circuits that are small enough to be widely deployed. We find that through the combined use of the wearout monitor and stability detectors, we can efficiently and accurately manage the reliability of a processor, and re–coup the performance of a processor that would otherwise be lost when processors are over–provisioned to meet an expected lifetime. We simulate a 16 core DSP with a wearout monitor and stability detectors on a mix of four different media algorithms. Using the wearout monitor and stability detectors, we find that by reducing average performance by only 5%, we can increase the lifetime of the processor by 46%.

Keywords

Reliability, Wear-out

1. INTRODUCTION

Transistor scaling has yielded unprecedented performance gains for modern processors. However, similar reductions in supply voltage have not been possible due to the need to limit power consumption drawn by leakage current. By not reducing the supply voltage, we find the that the electric fields of the transistors are often stronger than required, that the current densities of the metal wires on our processors are higher, and the operating temperature of our processors is undesirable. This combination of strong electric fields, high current densities and high temperatures have caused concern that the transistors of our processors will wear out over time.

Wearout is not only a problem that impacts the reliability of a processor, but is also an effect that can impact the performance of a processor because as transistors wear out, they also become slower. To determine the clock period of the processor, the designer of the processor must first find the delay through the critical path logic for a voltage, temperature and process technology corner. Die-to-die and intra-die process variations also need to be accounted for when setting the clock period of a processor. On top of this, the designer needs to provision some added delay in a clock cycle to allow for wearout.

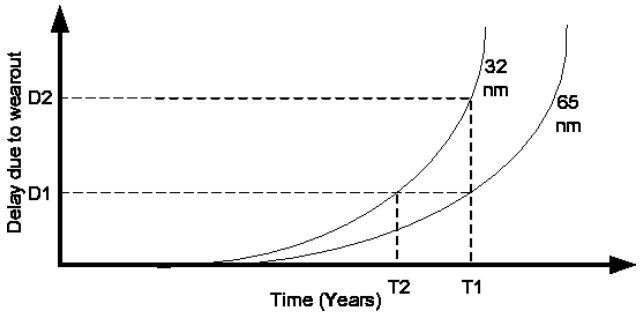

Figure 1: Shows the relationship between the lifetime of a processor and the amount of extra delay for processors built with two different circuit technologies.

We demonstrate the impact that wearout has on both the lifetime of a processor and the operation frequency of a processor using Figure 1. A processor manufacturer needs to guarantee a specific lifetime for a processor under some specified operating conditions. In technologies where circuit wear out is negligible, the additional delay that we need to pad each clock cycle with is also negligible. However, as we continue to shrink transistor geometries we find that wearout can become a problem in a time–frame where we are still interested in using the processor. As a consequence, to ensure the lifetime of a processor, the designer needs to make some assumptions about the rate of processor wearout. Then, the designer must pad the clock period with an amount of overhead in order to accommodate this wearout that happens over time. We refer to the

amount of clock–cycle padding required to meet a processor lifetime as $\Delta Wearout$. In Figure 1, for the 65 *nm* technology node, D1 is the amount of $\Delta Wearout$ that is required for a processor lifetime of T1. If we use the same amount of $\Delta Wearout$ for the 65*nm* curve and the 32*nm* curve, we find that the lifetime of the 32*nm* processor is now T2, where T2 is smaller than T1. On the other hand, if we wanted to preserve T1 as the lifetime of our processor in 32 *nm*, we would need to add D2 $\Delta Wearout$ to the clock cycle time of our processor.

What are the consequences of adding a constant $\Delta Wearout$ to the clock period of a processor? First, when the processor is new, and has no added latency due to wearout, the processor is actually running at a lower frequency than it is capable of. The processor is leaving some early–life performance "on the table" in order to ensure a given lifetime. This over-provisioning of the processor leads to a loss in possible performance while the processor is new. Second, eventually the processor will wear out where signals will no longer meet the clock cycle time (padded with $\Delta Wearout$). However, the processor could continue to operate at reduced frequency. The consequence is that we are losing some end–of–life performance.

One way to reclaim early-life and end-of-life performance is to make $\Delta Wearout$ represent the actual amount of wearout of the processor at any given time. We propose the use of stability detection circuits that can be used to find $\Delta Wearout$ for an individual processor. We could then have the freedom to operate the processor at higher frequencies than nominal during the early–life phase of a processor, or at lower frequencies during the end–of–life phase of a processor. On-line detection, rather than static computation of $\Delta Wearout$ is important for many reasons. First, it may be difficult to create models that accurately describe wearout because the physics of some wearout mechanisms are not well understood. Second, the companies that manufacture semiconductors may be reluctant to release detailed information about the wearout of transistors made using their fabrication process. Third, because of die–to–die and other manufacturing variations, configuring a static wearout model to a single manufactured processor is difficult.

However, simply re-claiming the performance during the early–life and end–of–life phase of a processor does not measure the remaining lifetime of a processor. So we are still left with the problem of ensuring a processor's lifetime that over-provisioning a processor's clock cycle time in the face of wearout had solved. To address this issue, we propose that a credit-system be used to account for the reliability of a processor. A processor is given a number of credits that is proportional to the manufacturer's guaranteed lifetime if that processor is operated at certain constraints (frequency, voltage, temperature, etc.). For every time unit we run the processor at the nominal frequency, we deduct some number of credits. If we continue to use credits at this rate and the number of reliability credits of the processor becomes very low, the probability of wearout of the processor occurring in the near future is high. We can run the processor at higher frequency than nominal, the processor uses more credits per time unit. Or, for every time unit the processor is idle, we will be saving credits that could be spent at later times of high processor demand. Likewise, a similar behavior is seen if the processor runs at higher or lower temperatures.

For instance, processor in a data center may be on an upgrade schedule where they are replaced by new processor every 3 years to do improvements in performance of new processors. Additionally, if we know that a processor will fail if run at maximum performance after two years, it may be wiser to slightly reduce the processor's performance in order get the processor to last for 3 years.

There are two goals of this paper. The first is to address problem of ensuring an expected lifetime of a processor using a credit-based wearout monitor (WM) that allows an operating system or user to manage reliability-performance trade-offs of a processor. We show that a WM also allows the operating system or user to budget the remaining reliability of the processor in order to maximize the performance of the processor for an expected lifetime.

The second goal of this paper is to employ a method for accurately measuring the wear out of a processor. To measure $\Delta Wearout$, we employ a low-overhead stability detection (SD) circuit that can be widely deployed on a processor [1]. Accurate on–line measurements of the wearout of different structures on a processor allow us to capture the performance normally lost in a processor's early–life and end–of–life due to conservative allocation of $\Delta Wearout$. Additionally, the combination of the credit-based WM and SD circuitry allows us to improve the accuracy of the WM by dynamically updating the relative wearout of the processor.

The rest of this paper is organized as follows. In Section 2, we discuss wearout–related work and contrast what is novel about our contribution. Section 3 gives a brief overview of wearout mechanisms that we model in this work. We then introduce our credit-based WM in Section 4.1. The SD circuitry and it's benefits are detailed in Section 4.2. We evaluate the WM and SD in the context of a multi–core digital signal processor with a mesh interconnect in Section 5. Finally, in Section 6, we discuss future directions for reliability aware architectures and then conclude.

2. RELATED WORK

One early work on architectural–level techniques for mitigating processor wear out was published by Srinivasan et al. in 2004 [2] [3]. This work describes a detailed wearout model of the processor to quantify the impact of scaling on lifetime reliability. They find that a 65*nm* design is over 3 times as likely to fail as a similar design in 180*nm*, and that time-dependent dielectric breakdown and electromigration are the primary modes of wear out in geometries of 65*nm* and smaller.

Later in 2004, the same group published two architectural methods for mitigating wear out [4]. They propose a Dynamic Reliability Management (DRM) strategy that explores the potential benefit of architectural re-configuration and dynamic voltage and frequency scaling (DVS). Their solutions has a drawback of requiring a precise device wearout model (called RAMP) to effectively predict wear out. Wearout models are difficult to accurately build, especially because the physics of all the wearout mechanisms are not fully understood. It is also difficult to build RAMP-like wearout models accurately because silicon foundries are reluctant to release detailed information about their process technology. Finally, because of die-to-die variations, it will be increasingly difficult to accurately characterize the parameters of semiconductors.

Building off the HotSpot tool [5], previous work has been done to show how much lifetime can be saved by reducing the operation frequency of the processor [6] [7]. Our WM techniques is one way to implement a method for managing these performance–reliability trade-offs.

To motivate how difficult it is to build accurate models of the wear out of processor, Figure 2 demonstrates how crucial small changes in the failure modes of semiconductors can be. In this figure, the "SPEC–FP Avg." and "SPEC–Int Avg." bars represent the results for a POWER4 processor as presented in [8]. Using a RAMP-like wearout model, we decrease the activation energy of metal by 0.1 eV and also allow the thickness of gate oxide to vary by 20%. We assume that these changes only impact the electromigration and time-dependent dielectric breakdown wear out, and those two wearout mechanisms account for 78% of all possible

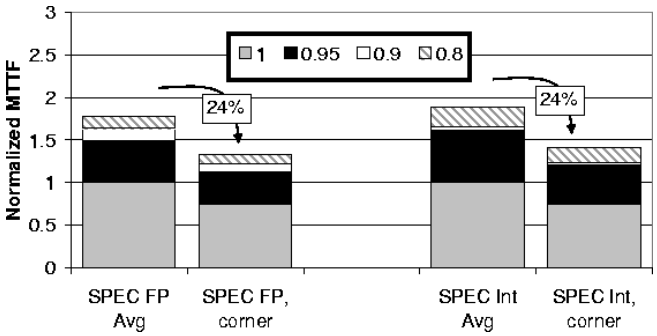

Figure 2: Demonstrates the impact of process variability on MTTF.

wearout modes [2]. From Figure 2, we can see that if this variability is introduced into the RAMP-like wearout model, the actual MTTF of the processor can be off by 24%, leading to pre-mature failure of the processor.

Instead of building static reliability models, we use stability detection circuits to accurately diagnose the level of wearout on the processor. This avoids the need for extremely accurate device wearout models. Additionally, since our stability detectors are used along side the datapath, they will experience the same activity as well as temperature as the circuit under test, providing an accurate assessment of the operating conditions of the processor.

Srinivasan et al. also proposed using structural duplication as well as allowing some structures to degrade in performance over time [8]. However, some structures were not protected, such as the instruction fetch and decode mechanisms, due to their size. The primary drawback of sparing is that having redundant structures on-chip is often costly in terms of area. This would typically be true for large on-chip structures, such as on-chip inter-core interconnects and instruction decoders.

Recently, Blome et. al. [9] [10] proposed a wearout detection unit, which measures wear out using a similar technique to what we propose. Their wearout detection unit (WDU) is similar to our SD circuit, but does additional statistics tracking in hardware to help filter out transient errors. Unfortunately, due to the size (7.5 um^2 in 130nm for each signal monitored) of the wearout detection unit, it may be expensive to deploy on a wide scale. By comparison, the SD circuit that we employ is much smaller citefrancoVLSI1994, allowing broader deployment for better monitoring of wear out.

The WDU uses chains of inverters to determine the delay of a signal, and includes statistical tracking circuitry that may be better computed in software. The reduced area of the SD is accomplished through the use of a feedback patch and by pushing the responsibilities of filtering transient instabilities to the operating system (OS).

Another unique contribution that we propose is the use of a credit-based WM that exposes the remaining lifetime reliability of a processor to the OS. Neither the WDU work, nor the RAMP work propose a strategy on how to ensure an expected lifetime of a processor, or how to maximize the performance of a processor during an expected lifetime.

The Razor work [11] shows architectural and circuit techniques for using DVS efficiently. They use time delayed redundant latches to detect timing violations due to data dependencies to determine if lower voltages may be used. This delayed latch-comparison scheme is somewhat similar to our SD circuit, but with different goals. Razor is looking for instability events, due to data dependencies, that happen on a much finer time–scale in order to change

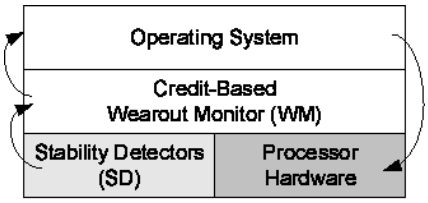

Figure 3: Overview of our lifetime management strategy.

the voltage to minimize power consumption. The SD circuits that we employ, are looking for wearout events that happen over a much longer time frame. The advantage of our stability detection system is that it is simpler as the Razor system provides some features that our technique does not need.

In summary, while our SD circuit is similar to the Razor and WDU, there are significant differences in the implementation of the SD with the Razor and WDU stability detectors. The RAMP wearout model is similar to the WM that we propose, but does not provide strategies for managing performance over the expected lifetime of a processor. Finally, the combination of the SD and WM is a unique contribution that provides both wearout detection and management.

3. WEAROUT MODELING

There are four wearout mechanisms that may result in increased delay in signaling that we simulate for this work. Electromigration (EM) is due to exchange of momentum between electrons and the metal atoms in wires, resulting in increase resistance, or in the critical case, a void. Time-dependent dielectric breakdown (TDDB) is the wear out of the insulating properties of silicon dioxide, which results in shifting threshold voltages. Hot carrier injection (HCI) is a phenomenon where either holes or electrons gain enough kinetic energy to embed themselves in the gate oxide or substrate of transistors. Negative bias temperature instability (NBTI) significantly shifts the threshold voltage of a transistor when under a constant electric field over time.

The wearout models for these failure mechanisms are similar to those used in [2, 8, 12] Additionally, similar to [2], we use a sum–of–failure–rates model to obtain the overall reliability of different computing structures. The constants that we use for the failure model are also borrowed from [2], and augmented where possible by values available in the ITRS roadmap [13]

The inputs to the wearout model that we use are: activity factor, frequency, voltage and temperature. To compute temperature, we use the default settings from HotSpot [5]. The outputs of the wearout model is the relative mean–time to failure (MTTF).

4. LIFETIME MANAGEMENT OF A PROCESSOR USING A WEAROUT MONITOR (WM) AND STABILITY DETECTOR (SD) CIRCUITS

Now that we have described the wearout mechanisms that we use in this study, we will describe the lifetime management strategy that we propose in this paper. This lifetime management strategy consists of two main parts, the WM and the SD circuits. Their relationship to the processor hardware and the operating system are shown in Figure 3. The hardware is augmented with SD circuits that monitor the wear out of the processor hardware. The wearout informa-

tion of circuits is passed from the SD circuits to the WM. The WM, which is in software, tracks the operation history of a processor and implements lifetime management policies, as directed by the OS. Wearout information passed from the SD circuits to the WM can be used to update the lifetime management policies of status of the processor. The rest of this chapter will describe the operation of the WM, and the SD.

4.1 Managing Processor Lifetime Using a Credit-Based Wearout Monitor (WM)

Let us assume that a processor has been given a nominal operating frequency for a given expected lifetime. For instance, a semiconductor manufacturing company could claim that their processor can run at 2 GHz for about three years before failing. From this expected lifetime, we assign a number of credits that represents this lifetime of the processor. For illustrative purposes, lets suppose that for every day the processor can run at 2 GHz, we give the processor 1 reliability credit. If the processor can run for nearly 3 years at 2 GHz, the total number of credits would be approximately 1000. For every day we operate the processor at 2 GHz, we will deduct one credit per day. As the number of credits the processor has approaches zero, we know that the processor should be getting closer to failure due to wear out. Of course, simply because we have exhausted all of the available credits does not mean that the processor has worn out, as any given processor may be more or less reliable than the typical case. However, the number of remain credits at any given time is an indicator of how worn the processor may be and premature failure of a processor should be a sufficiently rare occurrence if the characterization of a processor is done well. In Section 4.2, we will explain the use of SD circuits that allow us to modify the value of credits in order to get an accurate measure of the lifetime of a processor.

Continuing with our example, most processors are not run at a constant voltage and frequency throughout their lifetime. For periods of time where the processor is running faster than nominal frequency and voltage, the processor will wear out more quickly. Therefore, processors that operate at higher voltage and frequency should consume more credits per unit time than the same processor running at a lower voltage and frequency. Likewise, if the processor is running slower or even idle, the amount of wear on the processor is much lower. A similar effect is used for cases where the processor operated at different temperatures.

As an example, let us assume that running at 3 GHz per day requires 3 credits per day, while running at 1 GHz per day requires only one-quarter of a credit. If we go back to our example processor with 1000 credits, this processor could operate at 3 GHz for 333 days, or 1 GHz for 4000 days, or any combination in-between.

We compute the relative wearout of running a processor at a given voltage and frequency using the failure models in Section 3 with the assumptions of a processor manufactured in a 32*nm* process technology and a 25 FO4 pipeline. This yields a credits-per unit time curve that is quadratically related to operation frequency and exponentially related to temperature.

Figure 4 shows the relative amounts of wearout for different operation frequencies. This figure shows curves for a processor operating at 400 K and 300 K temperature, and assigns a credit amount proportional to the amount of wearout that can be expected for a circuit under these operating conditions. We can see that the wearout of a processor running at 3 GHz is more than three times the wearout of the same processor running at 2 GHz, which in turn is about five times more wearout than the same processor running at 1 GHz (similar to the values we used in our example). Another way to view Figure 4 is that a processor running at 3 GHz has an

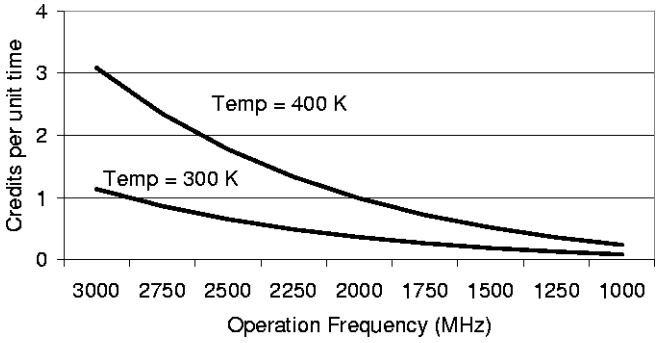

Figure 4: Shows the number of reliability credits required to operate a processor at different frequencies. Values have been normalized so that a 2 GHz operation frequency consumes one credit per time unit.

expected lifetime that is one–third of the same processor running at 2 GHz. Similarly, if the processor is running at higher temperature, wear out will be exacerbated, and the number of credits consumed per unit of time will be higher.

While it is possible to implement a WM in the hardware, we propose that the recording of credits be done in software. This should allow the OS to make some intelligent, wearout-aware decisions about how to use processor resources. Some examples of how the OS may use the wearout information of the processor include workload migration to avoid worn portions of a processor Another advantage of having the WM in software is that it could alert the user or administrator of processor components with dwindling reliability credits. The alternative is to maintain the credit values in hardware, which has the main advantage of less overhead. However, if the granularity of time unit at which credits are evaluated is sufficiently large (credits could be deducted only once per minute, for instance), we should be able to still get accurate representation of the processor usage while minimizing the amount of OS overhead.

4.1.1 Application of a Credit-Based Wearout Monitor (WM)

The WM lets the OS know the remaining lifetime of a processor through the accounting of reliability credits. With this knowledge, we may choose to run a processor at a frequency lower than its maximum in order to reduce the wearout on the processor and maximize the number of computations the processor can do within the lifetime desired from the processor. We call the desired lifetime of a processor (the amount of time before we replace the processor with a new processor), the "expected lifetime" of a processor. The expected lifetime of a processor is still subject to some minimum performance requirements, but that decision may be made by the user of the processor.

This trade-off between processor lifetime and performance is potentially useful in many computing applications. For example, a data center could potentially benefit financially by not having to replace worn processors and instead limiting the wear on older processors by limiting the frequency of these older processors. Meanwhile, a computer "gamer" may decide to run the newest computer game at the highest possible rate, and doesn't mind replacing their processor relatively quickly in order to maximize their enjoyment of the computer game.

Figure 5 shows the results of three example policies for credit use. The vertical axis of Figure 5 is the operation frequency, and the horizontal axis is time. The solid black bars in Figure 5 rep-

resent the application demand at any given time. The demand of the applications is generated randomly, and required the processor to run anywhere between 0 and 3 GHz to meet the demand of the application. Application demand may or may not be met by the processor, depending on the policy that governs the use of credits, and the availability of credits (remember, if there are no credits left, the processor has likely failed due to wearout). For this example, we again assume the credit system from Section 4.1 where the processor is capable of frequencies ranging up to 3 GHz (using 3 credits per time unit) and is at a constant temperature.

We test three different WM policies for managing the performance of a processor. The first policy is "DVS", which uses dynamic voltage scaling and is a greedy policy with respect to credits. This policy simply requests the necessary performance from the processor to match the demand of the application. The next policy, "Frequency Clipping", is similar to the "DVS" policy except that applications may only be granted frequencies up to 2.7 GHz. Finally is a "Linear Averaging" policy that tries to save credits for the end of the expected lifetime.

From the graph, we can see that all three policies continue to execute until a certain point. After that point, only the "Linear Averaging" policy has "saved" enough credits for the processor to continue to meet the demand of the application. The "Frequency Clipping" and "DVS" policies, since they are unaware of the expected lifetime of the processor, can not even partially meet the demand of the application during the last four time slots. Also of interest is that "Linear Averaging" out-performs the "DVS" policy by 8% in terms of maximizing the number of computations over the lifetime of the processor. Similarly, "Linear Averaging" outperforms "Frequency Clipping" by another 2% for this random application demand.

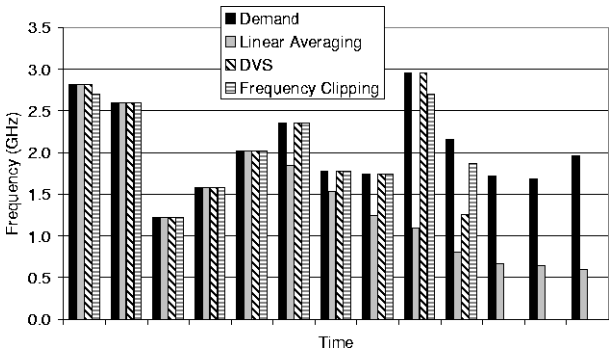

Figure 5: Shows the results of three different policies for using credits. The "Demand" bar shows the performance demands on the processor at any given time.

So far, the WM that we propose is similar to the reliability management strategy of RAMP [4]. The main difference is that the credit-system has knowledge of the in-use application demand, and can take advantage of that application demand to potentially save reliability for the end of the processor's expected lifetime. One limitation of this proposed WM, as well as RAMP is that they both assume that a processor's reliability can be accurately estimated during manufacture. There are several reasons why this may not be possible, that we mentioned in Section 2.

The consequence of inaccurate wearout modeling is that the credit-based WM could over-estimate the reliability of the processor, leading to premature failures, or under-estimate the reliability of the processor, leading to possible premature replacement of old processors. As we show in Figure 2, the consequences of small deviations in manufacturing parameters can have large impacts on lifetime. In Section 4.2, we address the issue of accurately measuring the wearout of a processor by proposing an on-line reliability detection system using SD circuits. This system allows us to dynamically "validate" the wearout model assumptions used in the WM which in turn allows for more accurate tracking the processor lifetime.

4.2 Measuring Wearout: Using Stability Detection (SD) to Calibrate the Wearout Monitor

It is difficult to accurately estimate the amount of wearout of circuits at design-time.Blome et al. [9] have recognized this shortcoming and implemented an on-line reliability measuring circuit that detects increases in latency using a circuit that measures increases in delay of a given signal. This circuit is called a wearout detection unit (WDU). The WDU is composed of two stages. The first stage, or the stability detection stage, samples the latency of signals through the use of multiple inverter chains. The second stage, or accounting stage, filters out transient events to find long-term trends of wearout.

The WDU is relatively large, and may only be deployed on a handful of signals. The result is that signals with particularly high activity levels could possibly be missed and fail, undetected. Instead, we propose using a lighter-weight stability detection circuit. We employ a similar methodology to the WDU, and push the filtering of transient events into software. Filtering in software should be sufficient because wearout-related events happen infrequently.

Instead of the inverter chains used in the WDU, our SD circuits use a structure similar to the circuitry in the Razor [11] project, where a signal is compared against a delayed version of the signal after a clock edge. If there is a difference between the clock-sampled signal and the delayed sampled signal, then the delay through the circuit has increased. While the Razor project uses this information to fine-tune voltage scaling in order to save power, we use this latency information as evidence of the wear out of a circuit. Section 4.2.1 will detail the stability detector design, and Section 4.2.2 explain how we use the stability detector to determine wearout. Finally, Section 4.2.3 will describe how the stability detection circuitry works in conjunction with the credit-based wearout monitor.

4.2.1 Stability Detection Circuitry

The stability detection circuit that we employ is borrowed from Franco and McCluskey [1] and is small enough to be employed on every signal on every stage in a pipelined processor. Figure 6 shows an example implementation of the SD circuit we use in this study. While this example is a master–slave latch, stability detection-enhanced clocking elements can be implemented for many clocking structures [14]. The total overhead is a single transistor, which is circled in Figure 6. In addition to latching the clocked signal, this circuit continuously compares the input signal "D" with a time–delayed copy of "D". When the clock is high and "D" does not match the value of the time–delayed version of "D", then that means the input signal has bit–flipped after the clock has transitioned. For this particular SD circuit, we can measure instabilities of about 2 FO1 delays continuously until the next clock edge. In the event an instability is detected, the error signal is propagated to a counter register, which is accessible by the WM. Because of the small size of this stability detector, we believe that it could be widely deployed. The only additional area overhead is for routing of the ERROR_n signals, which we do not model here.

4.2.2 Accounting for Transient Events

One concern with using the SD circuits is that we are using the

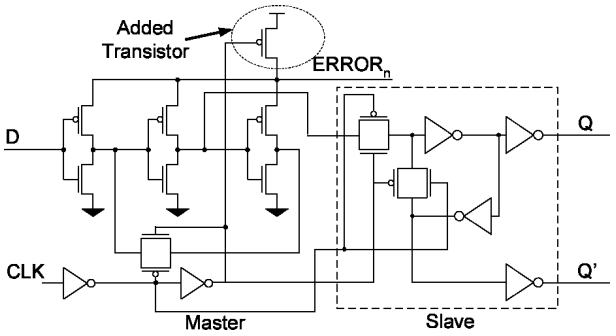

Figure 6: Example SD circuit which is built on top of a master–slave latch. Only a single transistor is added

Time Domain (s)	Mechanism
10^{12}	Lithography node
10^{9}	Electromigration
10^{8}	Hot Electron Effect
10^{6}	Negative bias temperature instability
10^{4}	Chip electrical mean variation
10^{-1}	Across-chip L_{poly} variation
10^{-4}	Self heating/temperature
10^{-8}	SOI history effect
10^{-10}	Supply voltage
10^{-10}	Line–to–line coupling
10^{-11}	Residual source/drain charge

Table 1: Time Scale of Variability in 65*nm* CMOS Devices [15].

SD to measure wear out, while other transient events could also trigger the SD. For example, if a processor is particularly hot, the latency of some signals on the processor may increase and trip our stability detector. Similarly, a fluctuation in the supply voltage may cause some signals to arrive late. How can we tell if the SD detects wear out or some other temporal event?

We propose that when the SD detects instability, we reduce the frequency and voltage of the processor momentarily. After a short amount of time, we can restore the voltage and frequency of the processor back to its original level. If we repeat this process several times, and the SD still detects instability, it is likely that the signal path's latency has increased due to wearout. This is because wearout is a long-term trend while transient events happen at a very short time scale. Table 4.2.2 shows the time scales of different common effects. A similar approach is used by Blome's WDU [9] circuit, which has statistical smoothing algorithms built into hardware in order to find long-term trends in signal stability.

The strategy employed to find the relative wearout of a processor through the use of SD circuits is an adaptive policy. This policy is shown in Figure 7. The first time the SD is triggered, we increment a record of how many times an instability has been detected. If the number of times the SD has been triggered exceeds a threshold, we will permanently reduce the operation frequency. This threshold is programmable in the WM. How much we reduce the operation frequency is also programmable. The assumption is that at the threshold, the signal has been measured to be slow enough times that the likelihood that the signal's delay has been increased by wearout is high.

On the other-hand, if the threshold of the number of instabilities has not been reached, we follow the top chain in Figure 7. The frequency of the processor is then reduced by one step. When the frequency is reduced, the old frequency is recorded by the WM so it can be restored at a later time. How large this reduction in frequency is again programmable by the WM. We then set a counter and decrement the counter after one time period has passed. If the counter has not reached zero, we simply wait another time period and decrement the counter by one. Once the counter has reached zero, we can then increase the frequency back to the previous operation frequency. This waiting period before raising the frequency should be small, as we only need to filter out transient events shown in Table 4.2.2. We assume, that from a lifetime perspective, that the performance due to the filtering of transient events is negligible.

We have now described the implementation of the SD and how we can use the SD to detect wearout. Section 4.2.3 will now describe how the combination of the SD and the MW allow us to manage the lifetime of processors accurately.

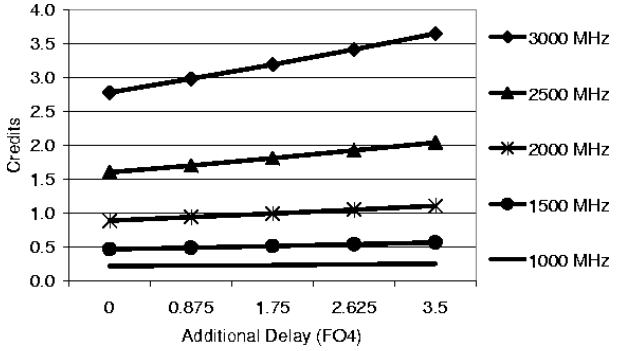

Figure 8: As increases in signal latency are detected by the stability detection circuit, the amount of credits required per unit time to operate at a given frequency increases.

4.2.3 Using the Stability Detection Circuit to Correct the Wearout Monitor

By using the stability detector, we can measure the increase in latency that is an indicator of wearout of the circuit that we are monitoring. If a circuit's latency increases, but we want to keep the circuit running at the same frequency, we can increase the supply voltage. This would allow us to maintain performance in the face of wearout. However, because the voltage is now increased, we are putting further stress on the worn system for running at the same frequency as a non-worn system. The result is that the rate of wearout is higher for a worn circuit running at the same delay as a new circuit.

Using the wearout measurements of the SD, we then propagate this information to the WM. The WM monitor in turn then needs to inflate the value of credits for operating at a given frequency because of the wear out measured by the SD. In this way, we are continually updating the the WM of the status of the hardware. If the processor happened to wear out early, the WM will detect it and in response, increase the number of credits to run at a given frequency earlier in the processor's lifetime.

Figure 8 shows the growth of credits for five different operation frequencies. We have assumed for this figure a temperature of 400 K. The data for this figure is created by first varying the amount of wear out for a circuit, as shown on the horizontal axis. Given this additional delay caused by wearout and a fixed operation frequency (each line represents a different operation frequency), we can com-

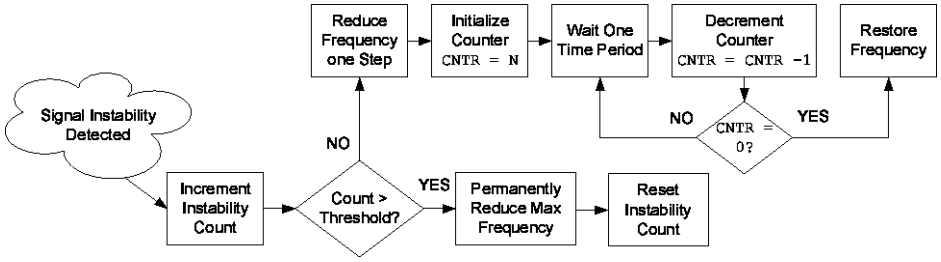

Figure 7: Policy for filtering out transient events that may trigger the stability detection circuitry.

pute the change in voltage required to maintain that frequency. The information of the voltage is then fed back into our wearout models from Section 3. Finally, in a similar manner to Figure 4, we normalize the wearout assign a relative number credits, depending on how the rate of wearout of the worn circuit compares with a new circuit.

5. APPLYING CREDIT-BASED WEAROUT MANAGEMENT TO A MULTICORE INTERCONNECT

In this section, we apply our credit-based wearout monitor to a simulation of a multi-core processor with an on-chip mesh interconnect network. The purpose of this simulation is quantify the amount additional lifetime performance the WM provides. We consider a multi–core processor with 16 processor connected by a packet switched mesh interconnect network in 32*nm* process technology. Our cores are modeled as Blackfin DSPs. The area estimate of the Blackfin core is 8.5 mm^2 in 130*nm* technology, scaled to 1.06 mm^2 in 32*nm*. The current consumption of our cores was estimated by using a combination of published power results for micro-architectural structures and VHDL synthesized using the Synopsys Design compiler in 130*nm* process technology. Through this estimate, we find that the Blackfin-based cores operate at 0.64 mW/MHz at a 1V supply in 130*nm*, which we scale to just over 0.8 mW/MHz in 32*nm*.

Interconnect transactions are assumed to be 256b and requests and acknowledges are 32b. We assume that the mesh is routed over the cores on metal layers M5–M6. Code is executed on Simplescalar [16] which is configured with similar parameters as a Blackfin DSP. The code trace is captured and turned into a data–flow graph, which then is partitioned iteratively using spectral bi-section using a tool called Chaco [17]. The partitioned graph is then simulated using a mesh simulator called FlexSim, written at USC [18]. FlexSim was configured in a 2-D space for up to 32 switches, where each switch is attached to an end-node with one injection channel to the switch. The default latencies were reduced to allow low-overhead flit-level routing as expected for an on-chip network. Routers are assumed to be 0.2 mm^2 in area. The overall area of this processor is 27.3 mm^2. For our inter-core interconnect power model, we employ power costs as abstracted from the Orion interconnect power model [19] from Princeton University. Link switching energy cost were taken from the "The Future of Wires" paper [20]. We find that our wires are using in the neighborhood or 10 pJ/bit for a 10mm trace, similar to Stanford's Smart Memories [21]. Given the floor-plan of the multi–core processor and the power consumption of the different units of the processor, we use HotSpot [5] to find the temperature of each of the units. We track the temperature of the cores, the routers, and the links of the processor using the Hotspot default parameters. A summary of the tool-chain used in this work is shown in Figure 9.

There are a few assumptions we make in this study. For the same temperature, frequency, voltage and activity level, we assume the same amount of wearout for processor cores, interconnect links and routers. However, different structures may be more or less susceptible to specific wearout types (the links are more likely to fail due to electromigration, for instance), this may not be strictly accurate. Since, the relative importance of the different failure modes in 32*nm* is not known, we perform a sensitivity analysis varying the failure rate of transistor–based wearout versus metal–based wearout. Regardless, the merits of the WM and the SD are independent of the source of wearout, as the SD circuits measure any wearout trend that results in increases in circuit latency.

We assume that we have SD circuits in every clocking element of the chip. The SD circuit as shown in Figure 6 has an area overhead of only one transistor, and that transistor should only be active very infrequently. So we assume that the overhead of the SD circuits is negligible. The routing overhead for the stability error signals we ignore in this study. We note that SD circuits may only need to be placed on signals that are likely to wearout, which is likely far less than the total number of latched signals on–chip, but we leave this for future research.

Another assumption we make is that a processor without a WM or SD is assumed to have a 20% $\Delta Wearout$ added to it's clock cycle time to accommodate 10 years of wearout at 2 GHz freqeuncy, 400K temperature. Finally, we assume that inactive circuits have negligible wearout. While leakage currents may induce wearout, we consider only active use of circuits as a first-order approximation of processor wearout.

We test this multi-core processor with a mix of four different media algorithms (FFT, Software Radio, Viterbi Decoder and MPEG4 encoding) that would be suitable to run on a multi-core signal processor. The aggregate workload we assume to have a peak variance of 60% over time, simulating varying demand placed on the processor. We assume that any processor that degrades to 20% of its orginal frequency is no longer usable. We test a processor with a WM with SD circuits on every clocking element using a linear averaging wearout policy on three different mixes of these applications against a processor without a WM.

Figure 10 shows the number of computations that our multi–core processor can perform over it's lifetime without and with the wearout prevention techniques proposed by this paper. There are three pairs of bars, the left most pair shows the results for a mix of applications, the middle pair show the results for a communications intensive mix, and the right–most shows the results for a computationally demanding mix of applications. The even mix of applications shows an increase of 32% of computations over the lifetime of the processor with a reduction of 5% in average

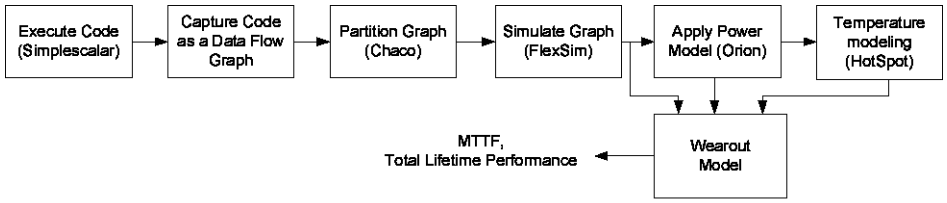

Figure 9: The tool–chain used for studying the impact of wearout on a multi–core processor.

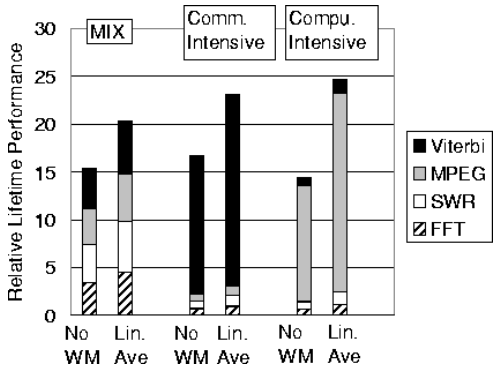

Figure 10: Lifetime performance of the processor cores on three different mixes of applications without and with the WM.

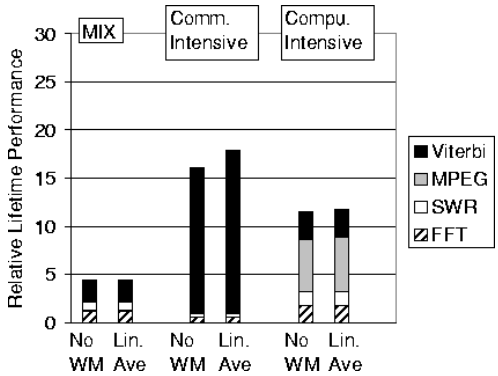

Figure 11: Lifetime performance of the interconnect on three different mixes of applications without and with the WM.

throughput. The reason for the drop in average throughput is the WM's linear averaging policy begins to limit the maximum performance of the processor in order to extend the lifetime of the processor. The communications-centric mix of applications shows a 37% increase in lifetime computations for a reduction of 6% in average throughput. Finally, the compute-intensive mix of applications shows only a 4% reduction in average throughput for an increase of 70% in lifetime computations. The aggregate result is that we can do 46% more computations for the life of the processor for a modest 5% reduction in performance. The reason some applications have longer lifetimes than others is due to how many stalls the applications have.

Similar to Figure 10 is Figure 11, which shows the relative lifetime performance of the interconnect network for the three different mixes of applications. There is very little improvement of the number of lifetime operations that can be performed by the interconnect with the WM versus without the WM. The primary reason for this is because our applications have almost twice as many computational cycles as communications cycles and therefore don't use nearly the entire reliability budget of the processor. Also, the mesh interconnect's path diversity helps considerably to spread the wearout across the interconnect. The interconnect's expected lifetime is 60.4% longer than the processor cores.

Now lets vary the amount of electromigration-related wearout. We now assume that electromigration accounts for 25% of wearout in the processor cores and 100% for the interconnect links. Additionally, we are going to double the rate of wearout of electromigration. Figure 12 shows the relative lifetime performance of the processor cores, and Figure 13 shows the relative lifetime of the interconnect. While similar to Figure 10 and Figure 11, there are a few noticeable differences. First, the overall lifetime performance has decreased for both the interconnect and the processor cores, with and without the WM. This is due to the fact that our processors are wearing out more quickly due to the higher rate of electromigration. Also, the difference in lifetime performance of the processor with and without the WM has increased. From this, we can conclude that the more prevalent wearout is, the more important it is to have a wearout management mechanism like the WM. Finally, Figure 13 shows that increasing the electromigration wearout impacts the lifetime performance of the interconnect more heavily than for the processors.

6. FUTURE WORK AND CONCLUSION

In this paper, we provide both a method for managing processor wearout, as well as a way to monitor wearout. Regardless if the processor is used in an application with high computational demands, like gaming, or large–scale throughput–based applications like data–centers, the combination of WM in conjunction with SD circuits allow the ability to accurately budget the reliable cycles of a processor as needed by the user.

While trading performance for lifetime is a viable way to extend the life of a processor, other techniques like structural duplication or structural enhancement provide complementary benefits that could also be incorporated into the WM. In the future, we plan on studying the combination of structural duplication, structural enhancement as well as performance management to further extend processor lifetime.

7. REFERENCES

[1] P. Franco and E. McCluskey, "On-line delay testing of digital circuits," in *Proceedings, 12th IEEE VLSI Test Symposium, 1994*, IEEE Computer Society, 1994.

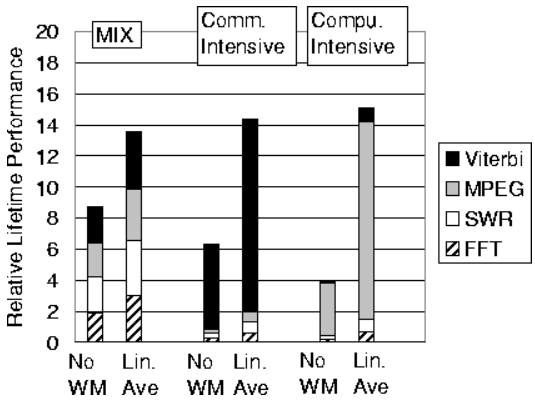

Figure 12: Lifetime performance of the processor cores on three different mixes of applications without and with the WM, with twice the amount of electromigration wearout.

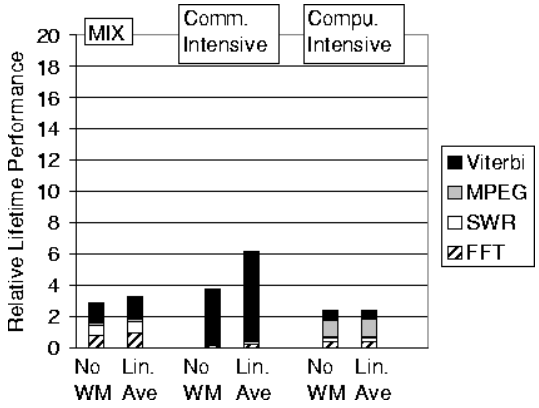

Figure 13: Lifetime performance of the interconnect on three different mixes of applications without and with the WM, with twice the amount of electromigration wearout

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

[3] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "A reliability odometer - lemon check your processor!," in *The Wild and Crazy Idea Session IV, in conjunction with ASPLOS XI*, 2004.

[4] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "The case for lifetime reliability-aware microprocessors," in *ISCA '04: Proceedings of the 31st annual international symposium on Computer architecture*, (Washington, DC, USA), p. 276, IEEE Computer Society, 2004.

[5] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *ISCA '03: Proceedings of the 30th annual international symposium on Computer architecture*, (New York, NY, USA), pp. 2–13, ACM Press, 2003.

[6] Z. Lu, J. Lach, M. R. Stan, and K. Skadron, "Improved thermal management with reliability banking," *IEEE Micro*, vol. 25, no. 6, pp. 40–49, 2005.

[7] Z. Lu, W. Huang, J. Lach, M. Stan, and K. Skadron, "Interconnect lifetime prediction under dynamic stress for reliability-aware design," in *ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, (Washington, DC, USA), pp. 327–334, IEEE Computer Society, 2004.

[8] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers, "Exploiting structural duplication for lifetime reliability enhancement," in *ISCA '05: Proceedings of the 32nd annual international symposium on Computer Architecture*, (Washington, DC, USA), pp. 520–531, IEEE Computer Society, 2005.

[9] J. Blome, S. Gupta, S. Feng, S. Mahlke, and D. Bradley, "Online timing analysis for wearout detection," in *The Second Workshop on Architectural Reliability (WAR), 2006.*, 2006.

[10] J. Blome, S. Feng, S. Gupta, and S. Mahlke, "Self calibrating online wearout detection," *MICRO 40: Proceedings of the 40th annual ACM/IEEE international symposium on Microarchitecture*, 2007.

[11] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," in *MICRO 36: Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, (Washington, DC, USA), p. 7, IEEE Computer Society, 2003.

[12] Joint Electron Device Engineering Council, "Failure mechanisms and models for semiconductor devices." www.jedec.org/ download/search/jep122C.pdf, 2006.

[13] ITRS, *International Technology Roadmap For Semiconductors - 2006 Edition, System Drivers*. Semiconductor Industry Association, 2006.

[14] P. Franco, "Testing digital circuits for timing failures by output waveform analysis," *Dissertation, Stanford University*, 1994.

[15] K. Bernstein, D. J. Frank, A. E. Gattiker, W. Haensch, B. L. Ji, S. R. Nassif, E. J. Nowak, D. J. Pearson, and N. J. Rohrer, "High-performance cmos variability in the 65-nm regime and beyond," *IBM J. Res. Dev.*, vol. 50, no. 4/5, pp. 433–449, 2006.

[16] D. C. Burger and T. M. Austin, "The simplescalar tool set, version 2.0," Technical Report CS-TR-1997-1342, University of Wisconsin, Madison, June 1997.

[17] B. Hendrickson and R. Leland, "The chaco user's guide, version 2.0, technical report sand94-2692," 1994. http://www.ti.com/ corp/docs/press/backgrounder/omap.shtml.

[18] U. SMART Interconnect Group, "Flexsim 1.2 flit level simulator." http://ceng.usc.edu/smart/tools.html.

[19] X. Chen and L.-S. Peh, "Leakage power modeling and optimization in interconnection networks," in *ISLPED '03: Proceedings of the 2003 international symposium on Low power electronics and design*, pp. 90–95, ACM Press, 2003.

[20] R. Ho, K. Mai, and M. Horowitz, "The future of wires," in *Proceedings of the IEEE*, vol. 89, pp. 490–504, April 2001.

[21] R. Ho, K. Mai, and M. Horowitz, "Efficient on-chip global interconnects," in *IEEE Symposium on VLSI Circuits*, June 2003. Stanford Univeristy.