A Self-Adaptive System Architecture to Address Transistor Aging

Omer Khan and Sandip Kundu
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA USA 01002
{okhan, kundu}@ecs.umass.edu

***Abstract*—As semiconductor manufacturing enters advanced nanometer design paradigm, aging and device wear-out related degradation is becoming a major concern. Negative Bias Temperature Instability (NBTI) is one of the main sources of device lifetime degradation. The severity of such degradation depends on the operation history of a chip in the field, including such characteristics as temperature and workloads. In this paper, we propose a system level reliability management scheme where a chip dynamically adjusts its own operating frequency and supply voltage over time as the device ages. Major benefits of the proposed approach are (i) increased performance due to reduced frequency guard banding in the factory and (ii) continuous field adjustments that take environmental operating conditions such as actual room temperature and the power supply tolerance into account. The greatest challenge in implementing such a scheme is to perform calibration without a tester. Much of this work is performed by a hypervisor like software with very little hardware assistance. This keeps both the hardware overhead and the system complexity low. This paper describes the entire system architecture including hardware and software components. Our simulation data indicates that under aggressive wear-out conditions, scheduling interval of days or weeks is sufficient to reconfigure and keep the system operational, thus the run time overhead for such adjustments is of no consequence at all.**

I. INTRODUCTION

The likelihood of device wear-out is a growing problem for advanced nanometer technology. International Technology Roadmap for Semiconductors (ITRS) states that "the development of semiconductor technology in the next 7 years will bring a broad set of reliability challenges at a pace that has not been seen in the last 30 years" [1]. The relentless pursuit of smaller geometries is approaching a point where technology limitations are pushing designs toward tighter constraints and expensive margins, elevating concerns about device availability and reliability [2]. The potential for these failures[1] decreases the expected lifetime of the processor, creating a lifetime reliability problem.

Processor lifetimes are traditionally managed through a combination of quality control metrics in manufacturing and conservative design parameters that reduce stress on a processor (e.g., running at a lower clock frequency and voltage to avoid high temperatures). Processors are typically designed with a mean-time-to-failure of 30 years, which assures few, if any units will fail during 11 years of "expected consumer use" assumed by manufacturers [3]. Scaling trends make it more expensive for quality control to meet this reliability goal, while conservative designs negatively impact performance.

Device aging has had a significant impact on transistor performance. Increased current density and temperature leads to faster degradation of transistors over time due to oxide wear out and hot-carrier degradation effects. Until 90nm technology, the degradation was small enough to be concealed by an upfront design margin in the product specification. But as the technology approaches 45nm and below, the worst case degradation is expected to become too large to be taken as an upfront design margin [2].

Product life acceleration with burn-in test is becoming less meaningful as well. To quote ITRS [1], "Two trends are forcing a dramatic change in the approach and methods for assuring product reliability. First, the gap between normal operating and accelerated test conditions is continuing to narrow, reducing the acceleration factors. Second, increased device complexity is making it impossible or prohibitively expensive to exercise or stimulate the product to obtain sufficient fault coverage in accelerated life tests. As a result, the efficiency and even the ability to meaningfully test reliability at the product level are rapidly diminishing."

Negative Bias Temperature Instability (NBTI) is a major source of device lifetime degradation [4]. NBTI affects PMOS transistors when the voltage at the gate is negative, causing the threshold voltage to increase. As a result both F_{MAX} and V_{MIN} of the design are impacted. The F_{MAX} is degraded because the circuits become slower over time, while memory structures experience an increase of their minimum voltage (V_{MIN}) to keep their contents.

Current practice is to use conservative frequency guard-bands of 10-20% to account for the performance loss due to device aging [5, 6]. For example, a device that clocks at 3GHz/1.1V during testing may be sold as a 2.7GHz/1.0V part to account for expected performance loss over product lifetime. This, in turn, requires designers to target for higher frequency of operation, thus significantly increasing power consumption [5][7].

The solution we propose avoids a large guard-band upfront, continually adjusting frequency and voltage over product lifetime. The main idea behind this scheme is to enable the system to adaptively adjust the operating frequency/voltage with minimal guard-bands to allow the system to operate at its peak

[1] Wear-out related failures, or intrinsic hard faults are distinct from extrinsic hard faults, which are permanent faults that result from manufacturing defects and are already present when a processor is tested in the factory. Thus, extrinsic hard faults are weeded out by testing. In contrast to extrinsic hard faults, the probability of intrinsic hard faults increases with long-term processor utilization. This paper addresses intrinsic hard faults.

978-3-9810801-5-5/DATE09 © 2009 EDAA

performance throughout its life. The adjustments are transparent to the operating system and application's software. This fine-grain management of device aging provides additional benefits of workload adaptation, runtime field testing, and non-stop system operation, which is not permissible in the conventional F_{MAX} or V_{MIN} testing that requires a tester.

The rest of the paper is organized as follows. The remainder of this section is devoted to providing some background and related work on NBTI and its impact on device reliability, followed by motivation for the proposed scheme. In section 2, we describe our proposed reliability management architecture. Section 3 and 4 provide our experimental methodology and data analysis. We conclude in section 5.

A. NBTI & Related Work

The severity of threshold voltage degradation due to NBTI depends on the operation history of a chip in the field: circuit parameters like operating frequency, supply voltage and temperature variance play a role, as well as data patterns due to variation in the workload characteristics. The workload determines the length of time a PMOS transistor may spend in ON state, when most of the performance degradation happens.

We have already mentioned why burn-in is losing effectiveness against NBTI problems [8]. Researchers have proposed solutions to mitigate NBTI by: reducing the amount of time the PMOS transistors observe a "0" at their gates [9]; resorting to classical redundancy techniques [10]; using software logging to handle crash detection and recovery [11]; using circuit and logic techniques to catch dynamic errors using special sequential circuits [11, 12]; and using runtime adaptation of the processor to changing application behavior, termed as Dynamic Reliability Management (DRM) [13, 14].

Although these techniques address the shortcomings of burn-in and guard-bands, they are either applied at a coarse-grain granularity or they require significant design cost overhead. For example, Razor DVS [15] proposes a technique to eliminate safety margins by running below critical voltage and subsequently tuning the processor voltage based on error rate. One of the main drawbacks of this work is the upfront additional circuitry required for Razor flip flops (RFF) and their associated power overhead. As RFFs are used on critical paths, meeting the chip's timing requirements and recovering pipeline state are challenging tasks that incur design overheads. On the other hand, DRM's uniform allocation provides high performance only for some applications, those that have high reliability slack, whereas our technique provides higher performance for all applications during the initial years and gracefully degrades performance as the device ages.

T. Austin et al., [16] propose a new software-based defect detection and diagnosis technique, which is based on using special firmware to insert tests for diagnosis, and if needed, repair through resource reconfiguration. Smolens et al., [17] present an in-field early wear-out fault detection scheme that relies on the Operating System to switch between functional and scan mode to test the chip in near-marginal conditions. Our technique uses similar software/hardware framework to address transistor aging, where the chip not only tests itself but also adapts to the changing conditions.

B. Motivation & Vision

The main drawback of burn-in and manufacturing time guard-bands is that they are static and expensive. Static guard-band may not be adequate for all parts; if the guard-band is increased, it may be excessive for other parts. This points to a need for flexible and scalable approaches that allow for continuous adjustments to combat degradation. The workloads running on a hardware platform are not static, but variable. The number of applications, their performance and power requirements, and the usage models vary based on the user demands and environmental conditions. Therefore, a continuous adjustment of frequency/voltage seems natural.

We propose a system level architecture that is based on virtualization of device aging management. Virtualization, in this context, is a software process with some hardware collateral that helps finding the optimal frequency. The proposed virtual framework provides architects with a layer of software that resides in the memory, concealed from all conventional software, thus isolating the functions of the implementation-specific device aging management features from the user and the operating system. The main idea is to expose the details of lower level hardware specific components to special software. This software provides flexible management capabilities of sensing, testing, and adapting the system over its lifetime. In an effort to address the drawbacks of conventional approaches discussed earlier, the proposed scheme has the following objectives:

Flexibility and Scalability: The layers of abstraction that exist between the hardware and software should hide intricate details that are necessary to manage frequency/voltage of the system efficiently and insulate the OS. This will allow hardware to evolve freely.

Low Cost: Frequency calibration without a tester will require some hardware collateral. This should be kept at bare minimum and should not impact power and performance of a processor.

Maximized Performance with non-stop management: Benefits from frequency adjustments will be greatest when the frequency decrements are small and adjustment is continuous.

Self and Field Testing: The proposed scheme allows the hardware to be its own instrument and enables self test during field operation. The flexibility of software allows the system to adapt to the changing environment and invoke the device aging management at variable intervals. Thus, if the device was controlling a Mars Rover, it will continue to adjust its operating frequency and voltage without requiring a tester attached to it.

Crash Recovery and Workload Adaptation: The proposed management software provides checkpoint capabilities to enable system recovery while the system tests itself. Additionally, the real-time environment and varying workload demands are used to optimize their effects on the lifetime reliability.

In summary the vision of the proposed virtual framework for device aging management is to adjust the system as performance degrades over its lifetime, and provide a cost effective and flexible solution that scales for future technologies.

II. System Reliability Manager

In this section we present the idea of a system reliability manager in the context of protection against device performance degradation caused by NBTI or similar physical causes. The core requirement for this manager is to sense the impact of power delivery, temperature and the workload on the hardware platform, and subsequently respond by reconfiguring the platform. The reconfiguration is primarily confined to the adaptation of supply voltage and/or operating frequency.

Pure hardware implementation of a reliability manager is costly and requires a priori information about the usage of a chip. On the other hand, pure software based approach needs instrumentation capabilities to address the issue of low level communication with the hardware. Additionally, operating system based implementation lacks flexibility due to strict interface abstractions to the hardware platform. These constraints drive us towards virtual management where the processor tests itself and finds its own frequency and voltage. An integral part of this system is crash recovery management that is built into the virtual layer.

The viability of a system reliability manager revolves around a cost-effective solution that can deliver self-testing and self-recovery capabilities in a flexible and scalable manner. In this section we describe this in detail. Our scheme has both hardware and software components. The hardware components are the knobs and their control mechanisms to adapt supply voltage and/or frequency to the changing reliability requirements [18]. The hardware platform also provides support for processor virtualization features like expanded isolation capabilities, and mechanisms for smooth and quick thread context switching capabilities [19].

The software component of our scheme is the device aging management software than runs natively as a guest privileged process on the hardware platform. We assume a thin Virtual Machine Monitor (VMM) running underneath the OS software stack, which is primarily used to enter and exit the System Reliability Manager (SRM) [19]. SRM software is concealed from all conventional software including the Operating System and may share the caching hierarchy of the platform for performance reasons. SRM software maintains a software timer for invocation control and crash recovery. SRM software also provides system checkpoint capabilities to enable self-testing capabilities without taking the system offline. Finally, the SRM software enables carefully crafted functional stress tests or built-in self-test control to identify degradation at a component granularity, and provides adjustments for sustained performance levels at target reliability. SRM software is akin to hypervisor that is commercially available [20].

A. SRM Architecture Framework

A high level system's view of the SRM architecture is shown in Figure 1. The SRM maintains a timer that is setup at chip initialization and then on every subsequent SRM exit. This timer is adjusted by the SRM to adjust its sampling to optimize the reliability requirements. When SRM is active, it has the highest privileged access to the hardware platform and the knobs to control supply voltage and operating frequency. The interface between SRM and the hardware platform is shown in Figure 2.

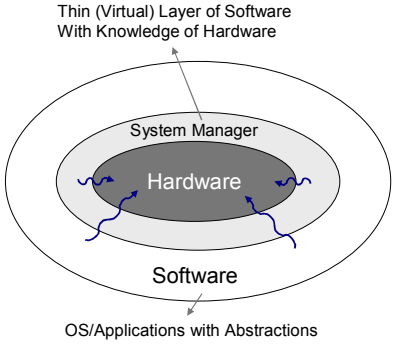

Figure 1. System Reliability Manager's System View

The Voltage Control Register (VCR) and the Frequency Control Register (FCR) are adjusted to control the hardware platform configuration. Once SRM software completes its work to determine the actions regarding device aging management, it exits via the VMM and passes control back to the Operating System. As a result, our approach delivers a hardware-software co-designed solution that assists the hardware to dynamically adjust to tackle the reliability concerns over the chip lifetime.

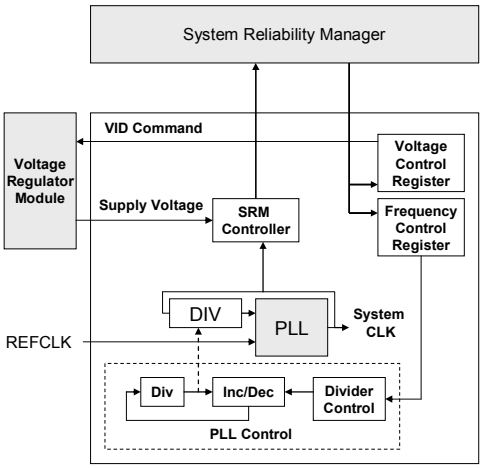

Figure 2. SRM Interface & Hardware View

B. SRM Software Flow

Figure 3 shows the flow diagram for the SRM software. Instead of using a worst-case guard-band over the entire lifetime of a design, the system starts off with the best-case frequency and voltage setting at first boot-up by invoking SRM. First invocation of SRM is specifically useful to calibrate a system to its power supply and cooling environment.

The steps for F_{MAX} testing are as follows:

i) Upon entry to SRM, all states are check pointed to ensure recovery from catastrophic system failure during testing. This includes the known operating F_{MAX}/V_{MIN} for system

ii) FCR is initialized to a low frequency value to set the frequency of the system. SRM timer is setup to enable self-recovery, and then test sequences are initiated

iii) If the test passes, FCR value is adjusted for a higher frequency, the timer is reset and test is rerun (back to step ii)

iv) If the test fails, upper limit on the frequency is found

v) If the system hangs, the timer interrupts. This interrupt automatically updates FCR to the last good value and passes control back to SRM for system recovery

Once the F_{MAX} is found for a given V_{DD}, the SRM adds a small guard-band to last until the next invocation of SRM. It also schedules the timer for next invocation of SRM and exits by giving control back to the OS. SRM can be invoked during subsequent boot-ups or by request from the system administrator. This is especially helpful when user/OS knowledge of the system's usage and load can be used to invoke re-evaluation of the chip. Additionally, SRM timer can be setup based on product specification or some on-chip degradation sensing mechanism. For example, NBTI, which is shown to have a large dependence on temperature can be analytically modeled in the SRM software, which can use the chip's thermal sensors to approximate the scheduling interval for re-evaluation. Additionally, if the system

is expected to degrade 10MHz every month, the SRM timer can also be statically set up to re-evaluate monthly.

Similar set of steps can also be used to find V_{MIN} for a given frequency. The information about V_{MIN} is critical for correct operation of Dynamic Voltage and Frequency Scaling (DVFS) for thermal management [21].

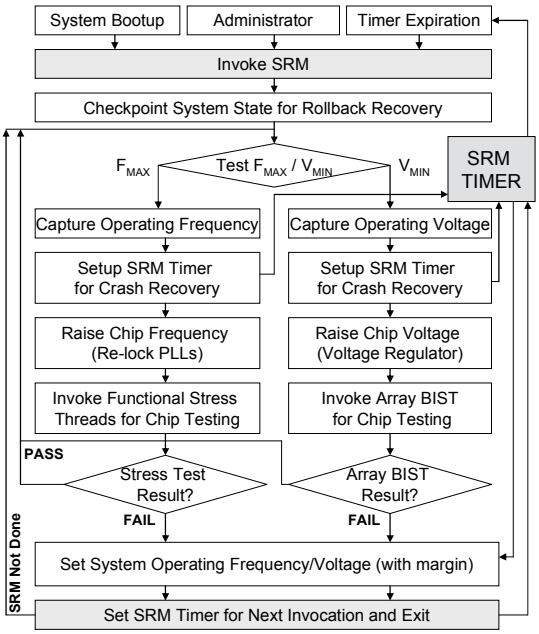

Figure 3. SRM Software Flow

C. Self-Testing Mechanisms

A key requirement for successful reconfiguration is complete knowledge about locations of failures and the nature of such failures. Our architecture framework offers low cost testing similar to the work presented in earlier research [16, 17, 22, 23]. Instead of relying on costly and time consuming built-in structures our software based scheme offers comprehensive functional testing framework. Based on our data analysis, presented in section 5, SRM is invoked at the granularity of weeks or days, so our methodology can use tests that run for longer durations (10s of ms). First phase for F_{MAX} testing involves carefully crafting software threads for the target system. In the second phase, these tests are compiled into SRM software, where code, data and exception handlers are setup along with routines for final result checking. During runtime, these test sequences are applied to the hardware platform as shown in Figure 3. Since these tests are run in the system environment unlike [23], the tests can make explicit external memory references.

Most of the modern designs come with lots of SRAM arrays. Due to a standardized structure of these arrays, built-in self-test (BIST) is commonly available on most designs with a diverse set of test vectors. Our framework provides a simple interface through SRM software to invoke these BIST engines and then check their results to determine a pass/fail for V_{MIN} testing.

D. Checkpoint and Crash Recovery

The main idea presented in this paper is to push the operating frequency and voltage to its limit, while the chip degrades during its lifetime. A major hurdle in such an architecture framework is that the system may crash during testing under such extreme operating conditions. The result of such a crash may range from incorrect results to a total system failure where a reset may be necessary. Our framework provides a cost-effective software-only mechanism to revert the system back to its pre-crash checkpoint of the system, similar to SafetyNet [24].

Whenever SRM is invoked to find the optimal operating frequency and voltage, a system-wide checkpoint is initiated. The checkpoint includes the state of the core registers, memory values and coherence/communication messages. The core registers are explicitly check-pointed, while the memory/coherence state is logged whenever an action (store or a transfer of ownership) might have to be undone. Additionally, all components in the chip are coordinated such that a consistent checkpoint is taken and stored in the non-volatile memory. Now the SRM can start its path finding process as shown in Figure 3.

In case the SRM is invoked due to a crash, the system rollback process is initiated. The cores restore their register checkpoints and the caches/memories unroll their local logs to recover the system to the consistent global state at the pre-crash recovery point. After the recovery, the system resumes execution. As the SRM invocation is done infrequently, the cost of taking a checkpoint and rollback is negligible considering that it's a one time cost for each SRM invocation.

E. Self-Recovery Knobs

The knobs needed to adjust F_{MAX} and V_{MIN} at runtime are shown in Figure 2. For operating frequency adjustment the new frequency setting can be adjusted by re-locking the PLL to the required setting. Additionally, the operating voltage is adjusted by sending a command to the voltage regulator module (VRM) to adjust the chip voltage. The VRM subsequently returns a new supply voltage. The SRM provides a simple interface to the hardware platform to request changes to the operating frequency and voltage.

III. EXPERIMENTAL METHODOLOGY

In this section we discuss our simulation environment. We use SESC cycle-level MIPS simulator for developing the SRM framework [25]. We have extended SESC to invoke Wattch [26] and Cacti [27] power estimation tools, and HotSpot temperature modeling tool [28]. For evaluating processor lifetime reliability at runtime, we integrated the RAMP model [29] in our simulator. Although RAMP provides analytical models for five intrinsic failure mechanisms, we only use NBTI in this study. We model a single superscalar processor with a floorplan containing twenty two structures. System parameters used are shown in TABLE I.

Processor Parameters	
Fetch, Issue, Retire Width	6, 4, 4 (out-of-order)
L1	64KB 4-way I & D, 2 cycles
L2	2M 8-way shared, 10 cycles
ROB Size, LSQ	152, 64
Off-chip memory latency	200 cycles
Hotspot Parameters	
Ambient Temperature	45°C
Package Thermal Resistance	0.8 K/W
Die Thickness	0.5 mm
Maximum Temperature	85°C
Temperature Sampling Interval	10,000 cycles
RAMP Parameters	
Qualification Temperature per Structure	82°C
RAMP Sampling Interval	10,000 cycles

TABLE I. System Parameters

The NBTI model used in RAMP is based on recent work by Zafar et al. at IBM [4]. This model shows that NBTI has a strong dependence on temperature in addition to electric field. The temperature and average MTTF is tracked for each structure in the processor over the entire simulation run. Our framework assumes that the first instance of any structure failing causes the entire processor to fail.

For our analysis, we chose SPEC2000 benchmarks. The choice of benchmark phases is primarily based on their thermal behavior with mcf being cold, gcc, gzip, ammp being moderate, and vortex, equake, art, bzip2 being hot. Each benchmark is fast forwarded 2 billion instructions, followed by HotSpot and RAMP initialization for each structure. This ensures that the processor as well as HotSpot and RAMP model get sufficient warm up.

We assume 65nm technology with chip wide maximum V_{DD} of 1.1V, and frequency of 2.0 GHz. For SRM evaluation, we vary Vdd and frequency by 5% downward steps up to a minimum of 0.88V and 1.6 GHz. For each benchmark, twenty five simulations are conducted with a pre-determined frequency/voltage setting. Each simulation is run for 1 billion instructions and performance evaluated based on throughput. At the end of each simulation, average MTTF per structure is sorted for each structure and the worst case MTTF is reported. For analysis purposes, we use an MTTF of 1 year, while we realize that expected consumer use for a processor is 11 years. Our results should hold for an 11 year MTTF as well.

IV. Results and Analysis

Figure 4. Figure 4 shows the impact of varying frequency at a given voltage setting for the hottest structure in the bzip2 benchmark. As the frequency is scaled down, the benchmark's performance degrades, while the temperature falls. On the other hand, Figure 5 shows that impact of voltage on temperature, assuming the chip remains functional. We assume that initially the chip is fully functional at 2.0GHz and 1.1V, which implies that for this voltage, frequencies below 2.0GHz are allowed. If the voltage is lowered, the maximum operating frequency will degrade. SRM on its invocation iteratively evaluates for the maximum possible operating frequency under a specified operational voltage. The key question is: What is the scheduling interval for SRM?

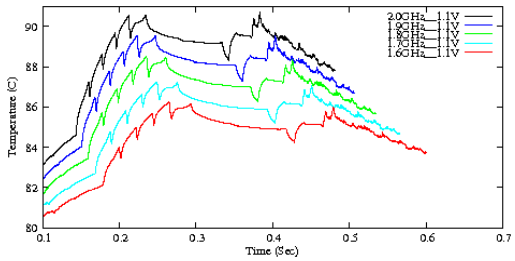

Figure 4. Scaling Freq. for bzip2: Thermal profile for IntReg

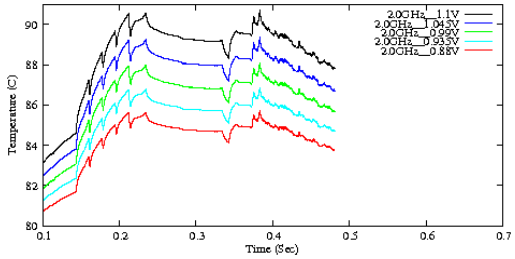

Figure 5. Scaling voltage for bzip2: Thermal profile for IntReg

Figure 6 plots the performance and the worst case MTTF for all benchmarks when simulation is run at 2GHz and 1.1V. It can be concluded from our simulations that the performance (IPC or throughput) of a benchmark directly impacts the worst case MTTF for the chip. How quickly one can expect a failure to occur is dependent on the workload. So, a mechanism that keeps track of the performance of each live thread in the system is desirable for tuning the scheduling algorithm for the SRM.

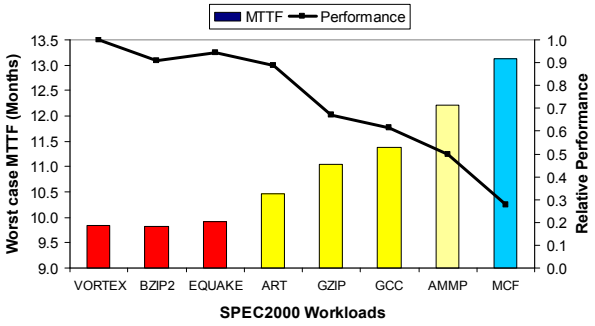

Figure 6. Lifetime Reliability Tradeoffs for SRM

Figure 7 shows the impact of the chip's lifetime degradation when frequency is scaled down. The rate of change in MTTF is linear and its slope is dependent on the type of benchmark. This data shows that workload's thermal and performance behavior can be used as a metric to track the rate of change in the MTTF.

We also observe from this data that even though we designed our system to sustain MTTF of 1 year with a qualification temperature of 82°C for each structure, the worst case MTTF can be better than expected. For example, mcf benchmark has the worst case temperature of ~80°C, which results in no expected degradation for the 1 year period. Hence, if the system only runs under similar workload conditions, the SRM scheduling is not needed for the 1 year period. On the other hand, if vortex or thermally hot workloads are being run on the system for the prescribed period, initially a monthly re-evaluation will suffice and once the system starts degrading, re-evaluation can be scheduled for twice a week.

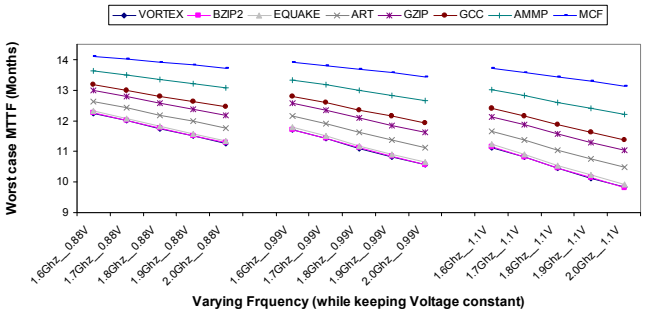

Figure 7. Impact of varying frequency across benchmarks

Figure 8 shows the worst case MTTF degradation for each benchmark under all operating conditions (sorted by MTTF) considered in this study. The x-axis shows that all benchmarks can achieve an MTTF of 1 year if the system constantly operates at 1.6GHz and 0.88V, but this comes at the cost of performance. On the other hand, if the operating conditions are initially set to 2GHz and 1.1V, the system can be operational for most of its lifetime. SRM can be invoked at regular intervals to adjust the frequency downwards and keep the system operational.

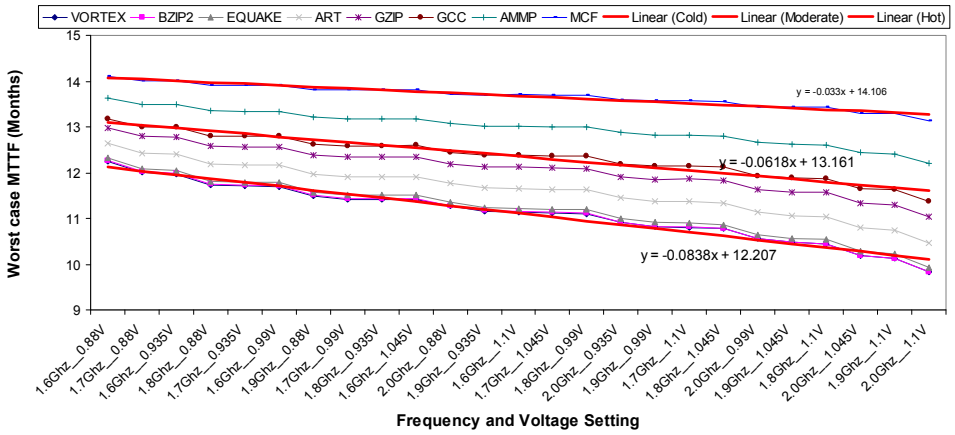

Figure 8. Rate of change for worst case MTTF as a function of benchmarks and operating conditios

V. CONCLUSIONS

We have presented a novel device aging management scheme for continuous adjustment of frequency and minimum supply voltage based on a co-designed virtual machine. The scheme requires no tester for determining F_{MAX} and V_{MIN}. Hardware collateral to implement this scheme is minimal and includes instructions for updating frequency and voltage control registers, which are already found in modern processor systems. The proposed solution allows the hardware to be its own instrument and enables self test during field operation by guiding the system to crash and recover during adjustment of its operating conditions. By insulating the device aging management from conventional software, the proposed framework shields the system and application software from managing low level details. The flexibility of software allows the system to adapt to the changing environment and invokes device aging management at appropriate intervals. The greatest benefits of this approach are (i) device operation near peak frequency throughout product life (ii) protection against failure due to insufficient lifetime guard-band, (iii) and no system downtime or change from a user perspective.

ACKNOWLEDGMENT

This work is supported in part by a grant from National Science Foundation. We thank thank David Albonesi and Paula Petrica of Cornell University, Sarita Adve of UIUC, and Pradip Bose of IBM Corporation for supporting the integration of RAMP reliability model.

REFERENCES

[1] International Technology Roadmap for Semiconductors (ITRS). Document available at *http://public/itrs.net/*

[2] S.Y. Borkar, "Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation", In IEEE Micro, Vol. 25, Issue 6, Nov.–Dec. 2005

[3] J. Srinivasan et al., "The case for lifetime reliability-aware microprocessors", In Int'l Symposium on Computer Architecture, June 2004

[4] S. Zafar et al., " A Model for Negative Bias Temperature Instability (NBTI) in Oxide and High K PFETs", In Symposia VLSI Technology and Circuits, 2004

[5] W. Abadeer, W. Ellis, "Behavior of NBTI under AC Dynamic Circuit Conditions", In Int'l Reliability Physics Symposium, 2003

[6] M. Agostinelli et al., "Erratic Fluctuations of SRAM Cache Vmin at the 90nm Process Technology Node", In Electron Devices Meeting (IEDM), 2005

[7] V. Reddy et al., "Impact of Negative Bias Temperature Instability on Digital Circuit Reliability", In Intl. Reliability Physics Symposium, 2002

[8] S. Kundu et al., "Trends in manufacturing test methods and their implications", In Intl. Test Conference, pp. 679-687, 2004

[9] J. Abella et al., "Penelope: The NBTI-Aware Processor", In Int'l Symposium on Microarchitecture, 2007

[10] S. Mitra, E. J. McClusky, "WORD VOTER: A New Voter Design for Triple Modular Redundancy Systems", In Symposium VLSI Test., 2000

[11] T. Lin et al., "Error Log Analysis: Statistical Modeling and Heuristic Trend Analysis", In IEEE Transactions on Reliability, Oct. 1990

[12] A. Tiwari et al., "ReCycle: pipeline adaptation to tolerate process variation", In Int'l Symposium on Computer Architecture, 2007

[13] J. Srinivasan et al., "Lifetime Reliability: Toward an Architectural Solution", In Int'l Symposium on Computer Architecture, May 2005

[14] J. Srinivasan et al., "The Impact of Technology Scaling on Lifetime Reliability", In Intl. Conference on Dependable Systems and Networks, 2004

[15] S. Das et al., "Razor: A self-tuning DVS processor using delay-error detection and correction", In Symposium on VLSI Circuits, 2005

[16] K. Constantinides, O. Mutlu, T. Austin, V. Bertacco, "Software-Based Online Detection of Hardware Defects: Mechanisms, Architectural Support and Evaluation", In Int'l Symp. on Microarchitecture, 2007

[17] J. Smolens et al., "Detecting Emerging Wearout Faults", In IEEE Workshop on Silicon Errors in Logic – System Effects, 2007

[18] J. Tschanz et al., "Adaptive frequency and biasing techniques for tolerance to dynamic temperature-voltage variations and aging", In Int'l Solid State Circuits Conference, 2007

[19] J. Smith, R. Nair, "Virtual Machines: Versatile Platforms for Systems and Processes", Morgan Kaufmann Pub, 2005

[20] "IBM Systems Virtualization", IBM Corp., Ver.2 Rel.1, 2005

[21] S. Naffziger et al., "Power and Temperature Control on a 90nm Itanium®-Family Processor", In Int'l Solid State Circuits Conference, 2005

[22] Y. Li, S. Makar, S. Mitra, "CASP: Concurrent Autonomous Chip Self-Test Using Stored Test Patterns", In Design, Automation and Test in Europe, 2008

[23] P. Parvathala et al., "FRITS – A Microprocessor Functional BIST Method", In Int'l Test Conference, 2002

[24] D. J. Sorin, et al., "SafetyNet: Improving the Availability of Shared Memory Multiprocessors with Global Checkpoint/Recovery", In Int'l Symposium on Computer Architecture, 2002

[25] J. Renau et al., "SESC Simulator", 2005; http://sesc.sourceforge.net

[26] D. Brooks et al., "Wattch: A framework for architectural-level power analysis and optimizations", In Int'l Symposium on Computer Architecture, 2000

[27] P. Shivakumar and N. P. Jouppi, "CACTI 3.0: An integrated cache timing, power, and area model", WRL Tech. Report, Compaq, 2001

[28] K. Skadron et al., "HotSpot: Techniques for Modeling Thermal Effects at the Processor-Architecture Level", In THERMINIC, 2002

[29] J. Srinivasan et al., "RAMP: A Model for Reliability Aware Microprocessor Design", IBM Research Report, Dec. 2003