Workload and User Experience-Aware Dynamic Reliability Management in Multicore Processors

Pietro Mercati
UCSD
pmercati@ucsd.eng.edu

Andrea Bartolini
University of Bologna
a.bartolini@unibo.it

Francesco Paterna
UCSD
fpaterna@ucsd.eng.edu

Tajana Simunic Rosing
UCSD
tajana@ucsd.edu

Luca Benini
University of Bologna
luca.benini@unibo.it

ABSTRACT

Reliability is a major concern for nanoscale CMOS circuits. Degradation phenomena such as Electromigration, Negative Bias Temperature Instability, Time Dependent Dielectric Breakdown worsen with transistor scaling. Dynamic Reliability Management (DRM) techniques reduce reliability loss at runtime by constraining operating points, but they face the challenge of reducing user experience degradation while meeting a lifetime target. In this work we propose a sensor based hierarchical controller for multicore processor DRM, exploiting the major gap between the time scales of workload variations and reliability loss. We improve performance and user experience by locally relaxing reliability-induced operating point constraints, while meeting them over the large time windows relevant for reliability. With respect to the state-of-the-art, our solution guarantees timely execution of 100% of latency-critical applications, and have a 4% performance improvement over the whole lifetime.

1. INTRODUCTION

Technology scaling has made modern integrated circuits more susceptible to degradation phenomena such as Negative Bias Temperature Instability (NBTI), Electromigration (EM) and Time Dependent Dielectric Breakdown (TDDB) [9]. Degradation depends on many process and environmental factors, but can be controlled by managing temperature and voltage. Degradation worsens under continued stress [18], while short spikes in temperature and voltage do not affect reliability much. Aging effects are described by mathematical models in terms of Mean Time To Failure (MTTF) [15] or reliability [20].

We focus on TDDB [6, 16, 20], but our solution can apply to other phenomena as well. TDDB is a degradation phenomenon that results in a low-impedance path through transistor gate dielectric, causing high leakage current that leads to failures. With the scaling of technology, increasing the design margin and binning the chips are becoming costly strategies. runtime management techniques can overcome this by dynamically changing chip operating points [15]. Modern processors can exploit dynamic voltage and frequency scaling (DVFS) to modify the degradation rate [18, 21]. Degradation can be estimated through voltage and temperature dependent mathematical models, but recent works also proposed devices that can directly sense degradation [13, 14] and are more accurate than temperature and voltage based estimation. In [14], TDDB-specific sensors are presented.

These elements, together with a control algorithm, are the basis for Dynamic Reliability Management (DRM). DRM has been proposed in [15, 10, 21] as a mechanism to trade off between system performance and reliability margin. DRM policies guarantee a target of reliability within the predefined lifetime of the system. However, the long time scale and the non-reversibility of degradation pose a challenge for the DRM control algorithm. To alleviate this issue, [21] proposes a predictive control for a single core CPU, based on a mathematical model.

Many systems today use multicore processors, from servers to smartphones and tablets. Embedded devices exploit multicore CPUs for data and computing intensive applications with varied requirements in terms of performance and QoS [1, 2, 8]. Android based systems exploit the mechanism of *intent* to describe and communicate to the hardware the urgency/quality of a task to be executed. Not satisfying these requirements causes significant user experience degradation. Little has been done for the reliability of multicore CPUs [5, 17, 19]. The main problem with state-of-the-art DRM solutions is that they control reliability by fixing a maximum limit of the operating conditions, disregarding the potential degradation of the user experience.

In this work we propose a novel DRM policy for multicore platforms. The proposed policy is based on a two-level controller, composed by a *Long Term Controller* and a *Short Term Controller*. The two levels operate on two different time scales, that we have called *Long Intervals*, corresponding to days that it takes for reliability to change, and *Short Intervals*, corresponding to OS scheduling ticks. Our controller monitors system reliability on a long time scale and adapts operating conditions to workload phase changes on a short time scale, with the goal of meeting a target reliability within a predefined target lifetime. Our solution uses aging sensors to improve reliability control robustness. The main novelty we introduce is the *Borrowing Strategy*, through which our solution is able to locally relax reliability-induced operating point constraints, while still meeting them over the large time windows relevant for reliability loss. This is a key feature for systems like smartphones and tablets, that emphasize user experience. We compare our policy against state-of-the-art and show that with our solution 100% of latency-critical applications meet their needs, with an overall performance improvement of 4% over the whole lifetime.

DAC '13 May 29 - June 07 2013, Austin, TX, USA

2. RELATED WORK

Traditionally chips are designed under the assumption of worst-case conditions [10]. This approach ignores the dynamic nature of actual operating conditions, which results in an overly conservative, performance-limiting device. Srinivasan et al. [15] first introduced DRM as a technique where the processor can dynamically respond to changing application behavior to maintain its lifetime reliability target, by dynamic voltage and frequency scaling (DVFS). This was a significant enhancement over previous worst-case reliability qualification methodologies. Blome et al. [4] extended the approach to monitor and control the impact on lifetime reliability, through thread scheduling and DVFS. The authors show how to leverage the slack between the typical degradation and the worst-case one to improve performance in periods of high peak demand. Karl et.al. [10] explored DRM using a systematic model to improve performance. Both [4, 10] assume that the future workload is equal to the previous one. Therefore they are very sensitive to sudden workload variations.

Zhuo et al. [20] proposed a process variation and temperature-aware oxide reliability model, which can estimate reliability from temperature and voltage history. In [21] the authors propose a DRM framework that extends this model to periodically predict the future value of reliability at the target lifetime. Based on the difference between the predicted reliability and the target one, the controller sets a maximum operating voltage. This approach is less sensitive to workload variations compared to previous works, because it exploits a confidence-based workload estimation. Since the policy sets the maximum voltage, it is not able to guarantee speed bursts for high performance demanding tasks, causing user experience degradation. Moreover, it is entirely model-based, relying only on temperature and voltage readings, and does not use sensors to estimate aging. Therefore it has high model uncertainty.

Singh et al. [14, 13] propose oxide degradation sensors and sensor-based DRM approach. Degradation sensors are non-intrusive monitors designed to be integrated in modern CMOS circuits. The authors have designed, manufactured and tested these devices. In the rest of this paper, we refer to them as degradation sensors or aging sensors. Monitoring with these sensors helps mitigate model inaccuracy. Since reliability models are based on stress measurements, a strictly model-based DRM policy tends to be very pessimistic [14].

The presented approaches for DRM ignore the fact that aging is observable on a large time scale, and that degradation is affected by average, rather than immediate stress. Therefore, they neglect the opportunity for short performance bursts to meet quality requirements of real applications. Furthermore, none of the previous work considers multicore platforms, a key component in most computing devices today.

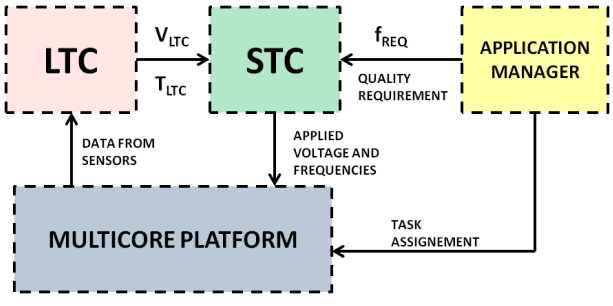

Figure 1: Two-level controller block diagram.

3. CONTROLLER ARCHITECTURE

The target platform is a homogeneous multiprocessor with N cores, with per-chip voltage setting and per-core frequency control. Each core is single threaded and has its own degradation sensors. Tasks are assigned in FIFO manner. The controller exploits voltage and frequency settings as knobs to trade off performance, while meeting the target temperature and reliability within a predefined lifetime. Figure 1 shows the basic building blocks of the proposed architecture, consisting of *Application Manager*, *Long Term Controller* and *Short Term Controller*.

Application Manager (AM): allocates tasks in FIFO manner, communicates the requested frequency f_{REQ} and the quality requirement for the task execution to the *Short Term Controller*. [1]

Long Term Controller (LTC): samples data from aging sensors at the beginning of each *Long Interval*, monitors the degradation status and calculates the average temperature and voltage. It predicts future reliability using these parameters. Since reliability loss occurs on a long time scale [15], we consider *Long Intervals* to be on the order of days. Based on the difference between predicted and target reliability, it computes a reference voltage, V_{LTC} and a reference temperature T_{LTC}, which are the inputs for the *Short Term Controller*. The constraint on reliability is met if the mean applied voltage in the *Long Interval* V_{LI} is less or equal to V_{LTC} and the temperature is below T_{LTC}.

Short Term Controller (STC): receives f_{REQ} and the quality requirement for the allocated tasks from the *Application Manager*, and V_{LTC} and T_{LTC} from the *Long Term Controller*. Based on that, it applies the *Borrowing Strategy*, adjusting voltage and frequencies at each scheduling tick given the tasks quality requirements, while keeping the mean applied voltage inside the *Long Interval* V_{LI} lower than V_{LTC}, and the temperature below T_{LTC}. The *Short Term Controller* can be coupled with state-of-the-art thermal management techniques to handle thermal emergencies [3, 12], given the thermal constraint from LTC. The operations performed by the blocks are discussed in more details in the following subsections.

3.1 Application Manager:

In modern operating systems, such as Android, the application can request a certain level of hardware and software service to provide a given QoS to the final user. Therefore, we characterize each task as either *Highly critical* (H) or *Less critical* (L) in terms of latency and user experience [2]. Executing H tasks at a frequency lower than f_{REQ} causes user experience degradation. This information allows the *Short Term Controller* to adjust its *Borrowing Strategy*. General purpose workloads for embedded devices do not contain profile information. Therefore the *Application Manager* selects $f_{REQ} = f_{MAX}$ for a running task and $f_{REQ} = f_{MIN}$ for the idle period.

3.2 Long Term Controller:

The diagram in Figure 2 shows the *Long Term Controller*. The *Long Term Controller* samples data from the aging sensors at the beginning of a new *Long Interval*, separately for each core.

The *sens2R* block estimates the reliability R_i for the i_{th} core, from the aging sensors readings S_i. R_i at time t is a number between $[0, 1]$ indicating the probability that the system will not fail before time t. It is a measure of the system degradation status [20]. For example, TDDB sensors, based on a ring oscillator whose fre-

[1] Our solution is orthogonal to the scheduler employed, which can be also of a different kind.

[2] Similarly, our solution could be adapted to real time systems, where the distinction between hard and soft deadline is tight.

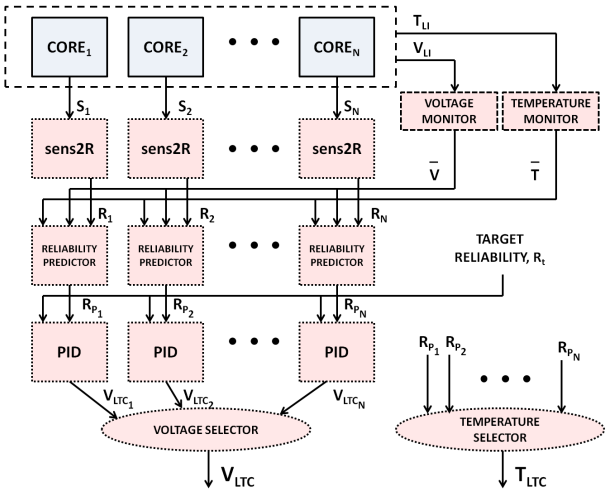

Figure 2: Long Term Controller block diagram.

quency increases as degradation takes place [13, 14], give statistically significant information on the aging status of their core [14]. In [13], authors map sensor readings for NBTI degradation to the system aging status. Based on that, they dynamically manage operating conditions to minimize NBTI degradation. In this work we assume that there exists a mapping between the output of TDDB sensors [14] and R_i and refer to papers [13, 14] for further discussion.

The *Voltage Monitor* and the *Temperature Monitor* keep the voltage and temperature history of the multicore platform by estimating the mean voltage/temperature that are going to be applied from the current time instant until the target lifetime. These values are $\bar{V}$ and $\bar{T}$, and they are calculated as an exponential moving average of the past voltages/temperatures, as:

$$\begin{aligned} \bar{V}_k &= \alpha_V \cdot \bar{V}_{k-1} - (1-\alpha_V) \cdot V_{LI_{k-1}} \\ \bar{T}_k &= \alpha_T \cdot \bar{T}_{k-1} - (1-\alpha_T) \cdot T_{LI_{k-1}} \end{aligned} \quad (1)$$

Where k identifies the k^{th} *Long Interval*, α_V and α_T are the weighting factors, $\bar{V}_{k-1}$ and $\bar{T}_{k-1}$ are the values at the previous *Long Interval*, $V_{LI_{k-1}}$ and $T_{LI_{k-1}}$ are the mean applied voltage/temperature in the previous *Long Interval*.

The *Reliability Predictor* receives R_i, $\bar{V}$,$\bar{T}$ and computes the predicted reliability R_{P_i} exploiting the model presented in [21]. R_{P_i} is the value of reliability that we would have at the target lifetime given a current reliability equal to R_i, and supposing that from the present time on we apply a voltage equal to $\bar{V}$ and a temperature equal to $\bar{T}$.

The *PID controller*, similarly as in [10, 21], receives R_{P_i} and the target reliability at the target lifetime R_t . Based on their difference, it calculates V_{LTC}, that asymptotically minimizes the tracking error. For the i^{th} core at the k^{th} *Long Interval*, we have $e_k = R_t - R_{P_k}$ and therefore:

$$\begin{aligned} V_{LTC_k} &= V_{LTC_{k-1}} + \\ &+ K_P \left(e_k + K_I \sum_{j=1}^{k} (e_j \cdot \Delta_{LI}) + K_D \left(\frac{e_k - e_{k-1}}{\Delta_{LI}} \right) \right) \end{aligned} \quad (2)$$

where K_P, K_I, K_D are the PID parameters and Δ_{LI} is the duration of a *Long Interval*. For example, if the system was subject to high temperature and voltage during the previous *Long Interval*, R_P will be low and the PID outputs a V_{LTC} lower than the previous one. If it was subject to low temperature and voltage, the PID outputs a V_{LTC} higher than the previous one.

The *Voltage Selector* selects the minimum V_{LTC}, outputted by the *Long Term Controller*, to guarantee the reliability of the most degraded core (the one with the lowest R_P). Only one V_{LTC} is needed[3], as the target platform only has a single voltage island. This work can be easily generalized to multiple V_{LTC}[4] Similarly, the *Temperature Selector* outputs the reference temperature T_{LTC}.

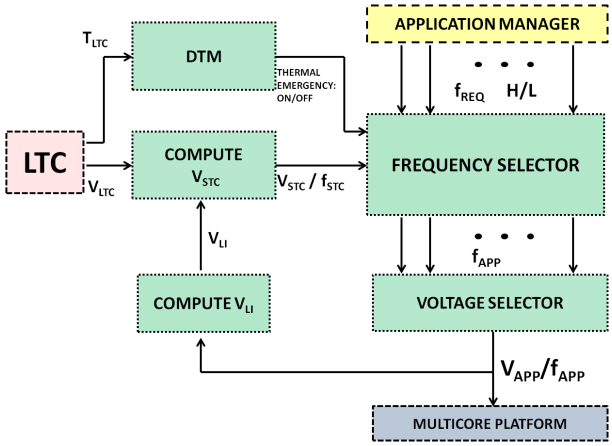

Figure 3: Short Term Controller block diagram.

3.3 Short Term Controller:

Figure 3 shows the block diagram of the *Short Term Controller*. The *Short Term Controller* receives V_{LTC} and T_{LTC} from the *Long Term Controller*, f_{REQ} and the H/L flags from the *Application Manager*. Based on these parameters, it develops the *Borrowing Strategy*, selecting the frequencies f_{APP} and the voltage V_{APP} to be applied at each *Short Interval* to meet the task quality requirements, while keeping V_{LI} less than or equal to V_{LTC}. Note that frequency selection should be coordinated with DTM. If temperature is higher than T_{LTC}, then a lower frequency is selected. Given that there has been a lot of work on DVFS for DTM, here we just focus on reliability aspects of voltage selection. The key to the *Borrowing Strategy* is the computation of the reference voltage V_{STC}:

$$V_{STC_l} = \frac{V_{LTC} \cdot \Delta_{LI} - V_{LI_{l-1}} \cdot t_{LI}}{\Delta_{LI} - t_{LI}} \quad (3)$$

Where l identifies the current *Short Interval*, V_{LI_l} is the mean voltage applied from the beginning of the *Long Interval*, and t_{LI} is the time elapsed since the beginning of the *Long Interval*. For each core executing a L task, the *Short Term Controller* selects $f_{APP} = f_{STC}$, while for each core executing a H task, it selects $f_{APP} = f_{MAX}$. For a idle core, it selects $f_{APP} = f_{MIN}$. Since the system has a single voltage island, in order to execute the most performance-heavy task, the controller selects the voltage V_{APP} corresponding to the maximum f_{APP}. [5] The other cores have the same applied voltage, but run at a lower or equal frequency. The

[3] In case of a platform with multiple voltage islands, cores can be grouped in voltage clusters. It would then be necessary to replicate this scheme for each cluster.

[4] Both the *Reliability Predictor* and the PID could be replaced by alternative solutions. This is an area of possible future improvement.

[5] In case of multiple voltage islands, it is necessary to replicate the scheme for each of them.

K_P	K_I	K_D	Δ_{LI}	α_V	α_T
$0.73[V/y]$	$14.4[y]$	$3.6[y]$	$25[d]$	0.1	0.1

Table 1: Simulation parameters

Borrowing Strategy is based on V_{STC}. If the controller applies a voltage lower than V_{STC}, V_{STC} tends to increase, allowing the system to go faster in the next intervals. If, conversely, a H task occurs and the controller applies a voltage higher than V_{STC}, V_{STC} tends to decrease, in order to "repay the loan".

$$V_{LI_{l+1}} = \frac{V_{LI_l} \cdot t_{LI} + V_{APP_l} \cdot \Delta_{SI_l}}{t_{LI} + \Delta_{SI_l}} \quad (4)$$

Where Δ_{SI_l} is the duration of the l^{th} *Short Interval*. As a *Short Interval* ends, V_{STC} is updated through Equation 3 and the *Short Term Controller* performs a new frequency/voltage selection. If $V_{LTC} - V_{LI}$ at the end of a *Long Interval* is non zero, the system has either not fully exploited the available reliability margin (if positive) or it has violated the reliability constraint for the current *Long Interval* (if negative). Therefore, this difference is added to the V_{LTC} which is computed for the next *Long Interval*, so to keep track of under/over-utilization.

4. RESULTS

The target platform is composed by 4 homogeneous cores with per-chip voltage and per-core frequency settings. The voltage ranges from 0.8V to 1.4V and the frequency ranges from 223MHz to 532Mhz. Our reference is the STMicroelectronics xSTsim architecture [11]. This platform is composed by a General-purpose Processing Element (GPE) acting as *host processor* and Processing Elements (PEs) acting as *streaming engine*. The GPE is an ST231 processor and the PEs are programmable processors with a simple ISA extended with SIMD and vector mode instructions. The platform addresses the needs of data-flow dominated, highly computational intensive tasks, typical of many embedded products.

For short term simulations we test our policy with xSTsim executing Inverse Discrete Cosine Transform (IDCT) [11] on a single *Long Interval* on random frame sequences. IDCT is a representative multimedia computational kernel, used in MPEG2 and JPEG decoding. Each frame is considered as a task. The GPE acts only as a dispatcher for the PEs and performs no computation. We have modified the application so to mark each frame as either H or as L, and to control the percentages of H and L tasks. The reliability control sets the operating voltage and frequency for the PEs by following the *Borrowing Strategy*.

For long term simulations we have developed a simulation infrastructure with Matlab, following the characteristics of the described platform. With this framework we can simulate the reliability model presented in [20] over the whole system lifetime and evaluate performance. The workload is modeled as a sequence of tasks with their own requested frequency and H/L flag. The task sequences are generated to reflect different user profiles [7], by varying the percentage of idle and busy periods and the percentage of H and L tasks. Table 1 shows the value of the parameters used in our simulations. The PID gains are obtained through Ziegler-Nichols open-loop method. The system is run at V_{MAX} for the entire lifetime and its response, e.g. the reliability curve, provides the parameters for the Z-N method. Δ_{LI} is set at 25 days, for having reasonable simulation times. α_V and α_T are both 0.1. This value has been chosen among others after extensive tests of the model with different workload and temperature profiles.

4.1 Comparison With State-of-the-Art

We compare our policy against the state-of-the-art technique presented in [21]. In this work, authors present a reliability management framework which uses the model in [20] as well. The framework periodically computes the predicted reliability R_P and exploit it to set a maximum operating voltage. This work has two main limitations:

- It limits the maximum voltage, causing user experience degradation when executing H tasks. We show how our policy, thanks to the two-level controller and the *Borrowing Strategy*, overcomes this limitation by following the task quality requirements, while still meeting the target reliability.
- It does not use aging sensors, and only exploits temperature and voltage readings for calculating reliability. We show how the use of aging sensors makes the reliability control more robust with respect to model variations.

Since the policy in [21] is for single core, comparison is conducted referring to this scenario. We assume the core to have a target reliability $R_t = 0.8$ and a target lifetime of 5 years. In the following, we denote our policy as *LTST* (*Long Term - Short Term*), and the policy in [21] as *Zhuo* (from the name of the author).

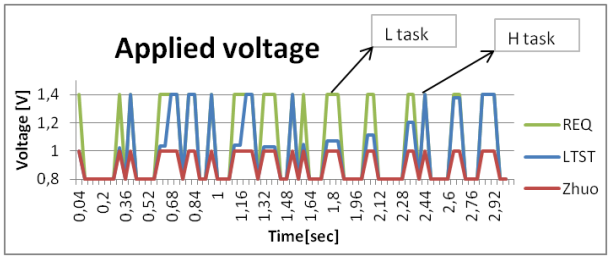

Figure 4: Voltage traces comparison on xSTsim simulator.

In Figure 4 we compare the voltage traces that we obtain with *LTST* and *Zhuo* for the execution of IDCT on a random sequence of frames in a *Long Interval*. V_{REQ} is $1.4V$ for a running task and $0.8V$ for an idle period. Executing tasks at a higher voltage allows to achieve better quality and higher performance. We assume that $V_{LTC} = 1V$. *Zhuo* is able to raise the voltage at most to $1V$. *LTST* achieves better performance for both H and L tasks. In the former case, *LTST* boosts voltage to V_{MAX}, and in the latter case, the voltage is set to V_{STC}, which is already higher than *Zhuo*'s 1V thanks to the *Borrowing Strategy*. Figure 5 shows the comparison in terms

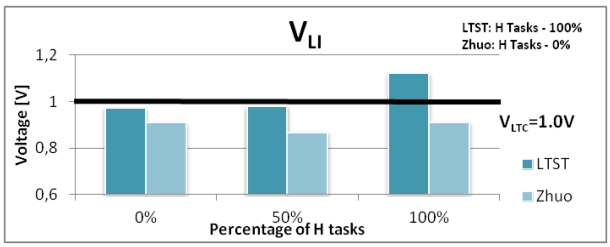

Figure 5: Mean applied voltage comparison on xSTsim simulator.

of V_{LI}. We show three cases, in which we vary the percentage of H tasks. In all of them *LTST* achieves a higher V_{LI}, and thus higher performance. In the cases of 0% and 50% of H tasks, *LTST* also keeps V_{LI} lower than V_{LTC}, respecting the reliability constraint for the current *Long Interval*. This means that *LTST* fully exploits the

available reliability margin, while *Zhuo* does not. In case of 100% of H tasks, the V_{LI} is higher than V_{LTC}. This is not a problem, since the *Borrowing Strategy* will add the difference $V_{LI} - V_{LTC}$ to the V_{LTC} of the next *Long Intervals*. By doing this, the next *Long Interval* will be slightly penalized to recover from this violation.

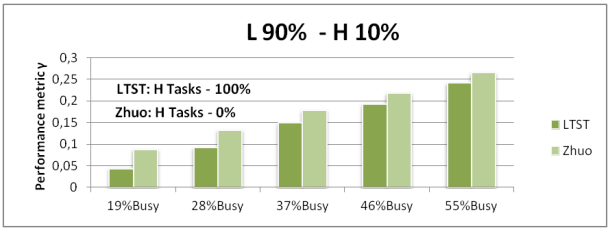

Figure 6: Performance comparison with Matlab long term simulator .

Figure 6 shows the comparison in terms of performance over the whole system lifetime, evaluated with the long term simulator. The comparison refers to different user profiles [7]. A higher percentage of *Busy* time, denotes a period of more intense user activity. For this evaluation we define a *Performance Metric* γ as:

$$\gamma = \frac{\sum_{i \in B}(f_{REQ_i} - f_{APP_i}) \cdot \Delta t_{B_i}}{\sum_{i \in B} f_{REQ_i} \cdot \Delta t_{B_i}} \cdot \frac{T_B}{T_B + T_I} \quad (5)$$

where B is the set of tasks (*Busy*), Δt_{B_i} is the duration of the task, T_B is the total *Busy* time, T_I is the total *Idle* time. Variable γ measures the frequency reduction with respect to the requested one for the executed tasks. Lower values of γ are better. Even if both solutions respect the target of reliability, Figure 6 shows that our policy presents a gain of 4% in terms of γ with respect to *Zhuo* on the entire lifetime. Moreover, our policy executes 100% of H task at their f_{REQ}, guaranteeing high quality execution to the final user. *Zhuo*, instead, slows down all the H tasks, causing significant user experience degradation.

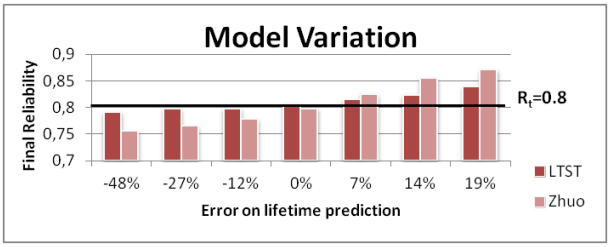

Figure 7: Robustness of sensor based approach.

To simulate the absence/presence of aging sensors, we distinguish the model that describes the real degradation M_{REAL} and the model that is adopted inside the reliability controller M_{CTRL} (through which R and R_P are computed). In case of *Zhuo*, no aging sensors are present. Therefore $M_{CTRL} \neq M_{REAL}$ and, as a consequence, the controller takes decisions based both on the wrong current reliability R and the wrong predicted reliability R_P. In *LTST* we have aging sensors that can give a better estimate of R. Therefore, only R_P is inaccurate. We define LT_{REAL} and LT_{CTRL} respectively as the lifetime obtained by controlling the system (with constant voltage and temperature) with M_{REAL} and M_{CTRL}. Therefore M_{CTRL} leads to an error on lifetime prediction, LT_{ERR}, equal to:

$$LT_{ERR} = \frac{LT_{CTRL} - LT_{REAL}}{LT_{REAL}} \quad (6)$$

LT_{ERR} measures the difference between M_{CTRL} and M_{REAL}. Figure 7 shows the comparison between *LTST* and *Zhuo* in terms of final reliability for different values of LT_{ERR}, expressed in percentages. For each case, *LTST* obtains a final reliability closer to the target with respect to *Zhuo* (final reliability is just 1% less than the target one with -48% LT_{ERR}). For this reason a sensors-based reliability control is significantly more robust.

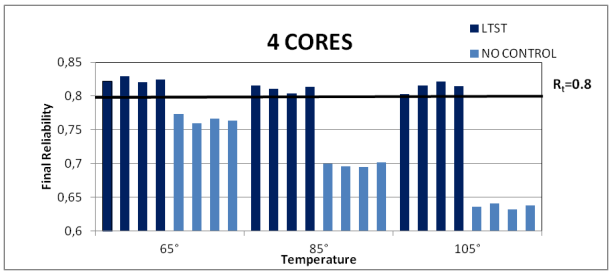

Figure 8: Robustness of the solution with temperature fluctuations

4.2 Multicore DRM:

in Figure 8 we show the benefits of adopting our solution for a platform with 4 cores, and its robustness with respect to long term temperature fluctuations. The target lifetime is 3 years, and the target reliability is 0.8. We set the nominal values of temperature T_{NOM} respectively at 65°, 85°, 105°. At each *Long Interval*, temperature is given by $T_{NOM} \pm U(-20, +20)$, where U is a uniform distribution, in order to simulate long term temperature fluctuation. For all the cases and cores, the target reliability is met and 100% of H tasks are guaranteed, whereas if control is disabled, the target reliability is violated. Figure 9 shows the reliability curve and temperature vs. time, for the 2nd core and T_{NOM} =85°. The controller easily meets the target target reliability of 0.8 by carefully managing the tradeoff between better performance (higher temperatures) and reliability.

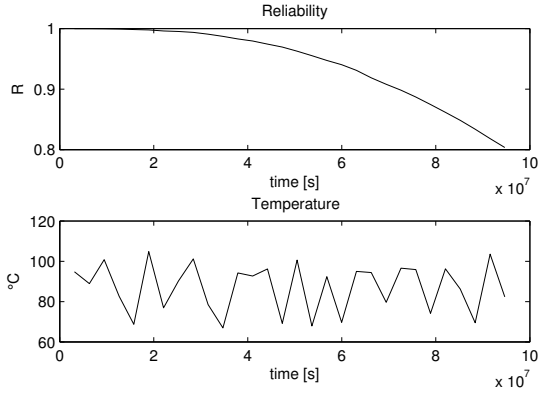

Figure 9: Reliability curve and long term temperature trace for one core.

5. CONCLUSION

In future technology scaling, many phenomena, such as TDDB, impact the system reliability. DRM techniques have been proposed to guarantee system lifetime, by constraining operating conditions, but they can cause user experience degradation. In this paper we propose a two-level controller: the first one acting on *Long Intervals* to set voltage and temperature constraints according to reli-

ability, and the second one which sets operating conditions over *Short Intervals* to meet quality requirement, while keeping the average voltage lower or equal than V_{LTC} and the temperature below T_{LTC}. Our solution uses aging sensors to improve reliability control robustness. We compare our policy against state-of-the-art and show that with our solution, 100% of latency-critical applications meet their needs, and show an overall performance improvement of 4% over the whole lifetime.

6. ACKNOWLEDGMENTS

This work was supported, in parts, by the EU FP7 ERC Project MULTITHERMAN (GA n. 291125) and the EU FP7 Project Phidias (GA n. 318013).

7. REFERENCES

[1] Apple imovie, itunes.apple.com/en/app/-imovie/id377298193?mt=8.

[2] ionroad, http://www.ionroad.com/.

[3] A. Bartolini, M. Cacciari, A. Tilli, and L. Benini. Thermal and energy management of high-performance multicores: Distributed and self-calibrating model-predictive controller. *IEEE Transactions on Parallel and Distributed Systems*, 24(1):170–183, 2013.

[4] J. Blome, S. Feng, S. Gupta, and S. Mahlke. Self-calibrating online wearout detection. In *Microarchitecture, 2007. MICRO 2007. 40th Annual IEEE/ACM International Symposium on*.

[5] A. K. Coskun, R. Strong, D. M. Tullsen, and T. Simunic Rosing. Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, SIGMETRICS '09, pages 169–180, New York, NY, USA, 2009. ACM.

[6] R. Degraeve, N. Pangon, B. Kaczer, T. Nigam, G. Groeseneken, and A. Naem. Temperature acceleration of oxide breakdown and its impact on ultra-thin gate oxide reliability. In *VLSI Technology, 1999. Digest of Technical Papers. 1999 Symposium on*.

[7] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin. Diversity in smartphone usage. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, 2010.

[8] B. Gyselinckx, C. Van Hoof, J. Ryckaert, R. Yazicioglu, P. Fiorini, and V. Leonov. Human++: autonomous wireless sensors for body area networks. In *Custom Integrated Circuits Conference, 2005. Proceedings of the IEEE 2005*, pages 13 – 19, sept. 2005.

[9] C. Hu. Gate oxide scaling limits and projection. In *Electron Devices Meeting, 1996. IEDM '96., International*, pages 319 –322, dec. 1996.

[10] E. Karl, D. Blaauw, D. Sylvester, and T. Mudge. Reliability modeling and management in dynamic microprocessor-based systems. In *Design Automation Conference, 2006 43rd ACM/IEEE*.

[11] F. Paterna, A. Acquaviva, A. Caprara, F. Papariello, G. Desoli, and L. Benini. Variability-aware task allocation for energy-efficient quality of service provisioning in embedded streaming multimedia applications. *Computers, IEEE Transactions on*, 2012.

[12] S. Sharifi, R. Ayoub, and T. Rosing. Tempomp: Integrated prediction and management of temperature in heterogeneous mpsocs. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pages 593 –598, march 2012.

[13] P. Singh, E. Karl, D. Blaauw, and D. Sylvester. Compact degradation sensors for monitoring nbti and oxide degradation. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 2012.

[14] P. Singh, E. Karl, D. Sylvester, and D. Blaauw. Dynamic nbti management using a 45 nm multi-degradation sensor. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 58(9):2026 –2037, sept. 2011.

[15] J. Srinivasan, S. Adve, P. Bose, and J. Rivers. The case for lifetime reliability-aware microprocessors. In *Computer Architecture, 2004. Proceedings. 31st Annual International Symposium on*.

[16] J. Stathis. Physical and predictive models of ultra thin oxide reliability in cmos devices and circuits. In *Reliability Physics Symposium, 2001. Proceedings. 39th Annual. 2001 IEEE International*.

[17] S. Wang and J.-J. Chen. Thermal-aware lifetime reliability in multicore systems. In *Quality Electronic Design (ISQED), 2010 11th International Symposium on*, pages 399 –405, march 2010.

[18] E. Wu, D. Harmon, and L.-K. Han. Interrelationship of voltage and temperature dependence of oxide breakdown for ultrathin oxides. *Electron Device Letters, IEEE*, 21(7):362 –364, july 2000.

[19] M. yu Hsieh. A scalable simulation framework for evaluating thermal management techniques and the lifetime reliability of multithreaded multicore systems. In *Green Computing Conference and Workshops (IGCC), 2011 International*, pages 1 –6, july 2011.

[20] C. Zhuo, K. Chopra, D. Sylvester, and D. Blaauw. Process variation and temperature-aware full chip oxide breakdown reliability analysis. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 2011.

[21] C. Zhuo, D. Sylvester, and D. Blaauw. Process variation and temperature-aware reliability management. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*.