# Improving Lifetime of Multicore Soft Real-Time Systems through Global Utilization Control

Yue Ma[1], Thidapat Chantem[2], Robert P. Dick[3], X. Sharon Hu[1],
[1]Department of CSE, University of Notre Dame, Notre Dame, IN 46656
[2]Department of ECE, Utah State University, Logan, UT 84322
[3]Department of EECS, University of Michigan, Ann Arbor, MI 48109
Email: {yma1, shu}@nd.edu, tam.chantem@usu.edu, dickrp@umich.edu

## ABSTRACT

System lifetime reliability is an important design consideration for many real-time embedded systems. Increasing integrated circuit power density and the subsequent rise in chip temperature negatively impact the lifetime reliability of such systems. Although existing thermal-aware methods are effective in reducing temperature, they cannot increase, and may even hamper, the system lifetime reliability. The complicated relationship between temperature and system lifetime requires that reliability be considered explicitly during system design. This paper presents a reliability-aware utilization control framework for homogeneous multicore soft real-time systems. The framework employs a model predictive controller to increase the system lifetime by manipulating the utilization of real-time tasks. An online heuristic algorithm is introduced to adjust the controller's sampling window in order to reduce the effects of thermal cycling on reliability. Simulation results show that the proposed approach can improve the system mean time to failure by at least 43% and as much as 369% compared to existing techniques.

## Categories and Subject Descriptors

B.8.2 [**PERFORMANCE AND RELIABILITY**]: Performance Analysis and Design Aids

## Keywords

System-level design; Reliability optimization; Dynamic voltage and frequency scaling

## 1. INTRODUCTION

Multicore systems provide high performance and power efficiency. However, due to CMOS technology scaling, multicore chips increasingly have higher power density and temperature, which, in turn, reduces system lifetime [1].

There has been a number of research efforts in increasing system lifetime by controlling temperature (e.g., [2–4]). However, since devices' lifetime is dependent not only on temperature, but also on thermal cycling [1], temperature reduction strategies alone are suboptimal in maximizing system mean time to failure (MTTF) [5]. To explicitly consider lifetime reliability, a number of papers have proposed techniques to increase system MTTF [6–8]. Coskun et al. presented a reliability-aware job scheduling and power management approach for multicore systems [6]. However, their work does not consider real-time requirements. Based on wear sensors, Hartman et al. designed a run-time based task mapping algorithm to improve the MTTF of real-time systems [8]. Unfortunately, wear sensors are not yet widely available and can only detect a limit set of integrated circuit (IC) failure mechanisms [9]. Compared to these studies, our approach considers different IC-dominant failure mechanisms to improve the lifetime reliability of soft real-time systems.

In this paper, we aim to improve the lifetime reliability of homogeneous multicore systems without sacrificing real-time performance. Since system MTTF depends on both temperature and thermal cycling, we introduce a reliability-aware utilization control (RUC) framework that jointly optimizes these two factors at the same time. RUC is composed of a global utilization control (GUC) to reduce temperature and a sampling window control (SWC) to reduce thermal cycling. Our main contributions are as follows.

(i) We show that increasing utilization via decreasing execution speed reduces the runtime temperature. We exploit this observation to design a model predictive controller (MPC) which keeps the system utilization at a desired value (known as utilization set point) to improve the MTTF without causing more deadline misses. The design of MPC considers the real-time constraints as well as MTTF dependency on core temperatures.

(ii) An important parameter in MPC design is the sampling window length ($L_{sw}$) which impacts system reliability as it affects both core temperatures and thermal cycles. A larger sampling window reduces thermal cycling but increases core temperatures. To achieve the desired trade off between temperature and thermal cycles, we introduce a heuristic algorithm to dynamically adjust $L_{sw}$ of the MPC.

We conducted a large set of simulations to assess the effectiveness of our approach. Compared to existing temperature-aware and utilization control mechanisms, our proposed framework achieves, on average, 43% improvement in MTTF, and up to 369%, while allowing more task sets to satisfy real-time constraints.

## 2. PRELIMINARIES

We consider four main IC-dominant failure mechanisms in this paper: electromigration (EM), stress migration (SM), time dependent dielectric breakdown (TDDB), and thermal cycling (TC) [5]. Figs.1(a), (b), and (c) depict the MTTF

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

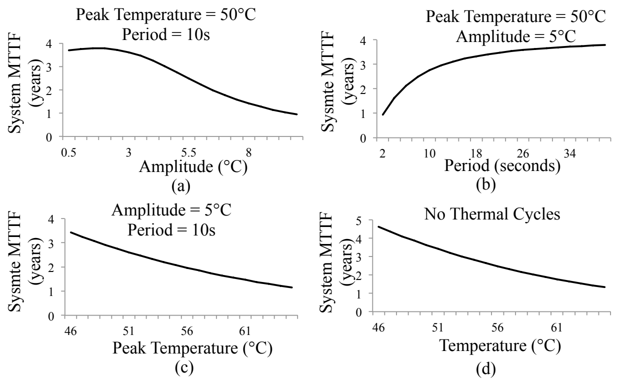

**Figure 1: System MTTF due to: (a) amplitude of thermal cycle; (b) period of thermal cycles; (c) peak temperature of thermal cycles; (d) temperature but no thermal cycles.**

of an example system as a function of amplitude (e.g., the difference between the highest (peak) and lowest (valley) temperature), period, and peak temperature, of thermal cycles, respectively, according to a reliability modeling tool [10] with default settings. For comparison purposes, Fig.1(d) shows the system MTTF due to temperature alone without thermal cycles. As can be seen from Fig.1, the system MTTF is generally higher with lower temperature and smaller thermal cycles, but the precise reliability model is somewhat more complicated.

We focus on homogeneous multicore systems in this paper. Let $M = \{\rho_1, \rho_2, ..., \rho_m\}$ denote the $m$ cores. We assume that these cores have identical thermal and electrical characteristics and initial wear state. The system MTTF of the multicore system is $\min\{MTTF$ of core $\rho_i\}, i = 1, ..., m$, and the MTTF of core $\rho_i$ ($MTTF_i$) can be calculated based on the core's failure rate, or using existing reliability modeling tools, e.g., [10]. For the multicore system under consideration, it has been proven that the system MTTF is maximized if the power is distributed evenly among the cores [9].

Power consumption is the fundamental cause of rising temperature and hence system failure. When a core performs operations, it dissipates dynamic ($P_{dyn}$) and leakage power ($P_{leak}$), but only consumes leakage power when it is idle [2]. Suppose the utilization of a core in a given time interval $\Delta t$ is $U = \frac{\Delta t_a}{\Delta t}$, where $\Delta t_a$ is the amount of time that the core executes operations, the average power ($\bar{P}$) can be calculated as

$$\begin{aligned}\bar{P} &= U \times P_{dyn} + P_{leak} \\ &= U \times V^{\alpha_0} \times f \times \alpha_1 + (\alpha_2(f,V) + \alpha_3(f,V) \times T) \times V,\end{aligned} \tag{1}$$

where $V$ and $f$ are the core's voltage and frequency, respectively. $\alpha_0 \geq 1$, $\alpha_1 > 0$, and $\alpha_2$, $\alpha_3$ are some voltage/frequency dependent parameters [2].

In term of workloads, we assume that tasks are periodic and already mapped to cores and that no migration is allowed. Workloads may change over time. Tasks on each core are scheduled by a real-time scheduling policy such as earliest deadline first (EDF) or rate monotone (RM) [11]. In a soft real-time system, the late completion of tasks is acceptable but should be avoided. The job of the $j^{th}$ task is denoted by $\tau_j$, which is associated with a tuple $\{e_j, d_j\}$ where $e_j$ is the execution time and $d_j$ is the relative deadline (as well as the period). For a given time duration, thereafter referred to as sampling window $k$ ($SW_k$), if core $\rho_i$'s frequency is fixed at $f_i(k)$, its utilization can be calculated as

$$U_i(k) = \sum_{\tau_j \in \Gamma(k)} \frac{e_j}{d_j} = \sum_{\tau_j \in \Gamma(k)} \frac{e_j^* + e_j'/f(k)}{d_j}. \tag{2}$$

where $e_j'$ reflects the portion of job execution that is dependent on the core's frequency, and $e_j^*$ is the independent portion. $\Gamma(k)$ is the set of jobs executed in $SW_k$.

## 3. PROBLEM FORMULATION

We will use a control theoretic approach to improve the system MTTF by controlling task utilization. Compared to temperature control and MTTF control, utilization control is easier to implement and already widely used in soft real-time systems [4, 12].

**Theorem 1.** *If in a given time duration, $SW_k$, the core temperature is constant, then a higher frequency/voltage pair used in $SW_k$ leads to a higher temperature but a lower utilization.*

Theorem 1 indicates that any method to control utilization through manipulations of core's frequency and utilization is also effective in temperature management. Before formulating the utilization control problem, we introduce some notations for the $SW_k$: $\bar{f}(k)$ is the average frequency over all cores; $U^s$ is the utilization set point; $U^{\max}$ is the upper bound on the utilization to ensure schedulability; $f_{\min}$ and $f_{\max}$ are the lowest and highest core's frequencies allowed, respectively. We aim to solve the following problem:

$$\min \sum_{\rho_i \in M} (U^s - U_i(k))^2. \tag{3}$$

The solution to (3) must satisfy the following constraints:

$$\begin{cases} U_i(k) \leq U^{\max} & \text{for } \rho_i \in M, \quad (4) \\ -f_{th} \leq f_i(k) - \bar{f}(k) \leq f_{th} & \text{for } \rho_i \in M, \quad (5) \\ f_{\min} \leq f_i(k) \leq f_{\max} & \text{for } \rho_i \in M. \quad (6) \end{cases}$$

The first constraint ensures no cores exceed the schedulability bound. Based on the system reliability definition given in Section 2 and [9], the second constraint is introduced to bound the differences in the cores' frequencies, which in turn bound the temperature differences among the cores. The third constraint is used to satisfy the cores' operating requirement.

## 4. DESIGN OF RUC

### 4.1 RUC framework

In this paper, we propose an online mechanism to improve system MTTF by solving the optimization problem in (3)-(6) and dynamically adjusting the length of sampling window. We refer to the overall framework as RUC. RUC consists of two main components: global utilization contrl (GUC) and sampling window control (SWC). GUC reduces temperature by using an MPC, which dynamically adjusts the cores' frequencies to enforce the utilization set point. SWC minimizes thermal cycles by dynamically adjusting $L_{sw}$.

At the end of $SW_k$, the utilization of each core is measured. Based on the measured utilization and MPC settings, MPC solves the optimization problem defined in (3)-(6). The solution consists of the cores' frequencies in the next sampling window and is sent to each core before the start of $SW_{k+1}$. At the end of $SW_k$, the measured temperature values are sent to and saved in SWC. Specifically, the temperature measured in the most recent $s$ sampling windows are saved. We refer to the $s$ sampling windows as one profiling window. At the end of each profiling window, a temperature profile is generated. SWC then analyzes this temperature profile and determines whether to increase or reduce the sampling window length in the next profiling window.

## 4.2 Global utilization control

In this section, we discuss our GUC design to solve the constrained optimization problem in (3)-(6) using an MPC. The basic idea behind any MPC is to optimize a cost function. Hence, we first make a comparison between the constrained optimization problem in (3)-(6) and an MPC cost function optimization problem. After this, we transform the MPC cost function optimization problem to a standard quadratic programming problem to be solved by some existing solvers.

According to control theory, the design of MPC is to minimize a cost function for a given $SW_k$,

$$J(k) = \sum_{i=N_1}^{N_2} \delta_i[U(k+i)-\rho(k+i)]^2 + \sum_{j=1}^{N_u} \lambda_j[\frac{1}{\Delta f(k+j-1)}]^2, \quad (7)$$

where $N_1$, $N_2$, $N_u$, $i$, and $j$ are integers, $\delta_i$ is the tracking error weight, and $\lambda_i$ is the control penalty error. The user specified reference trajectory $\rho(k+i)$ defines the ideal trajectory along which the utilization should converge to the set point. The first term in the cost function (7) is a variation of the function in (3). The second term in (7) minimizes the changes in the manipulated variable and does not affect the final result of the optimization problem. Hence, minimizing the cost function (7) under the constraints in (4)-(6) would also lead to the optimal solution to the problem defined in (3)-(6).

We propose using a quadratic programming solver to solve the optimization problem. A standard quadratic programming problem can be written as

$$\min_{\varepsilon}\{\frac{1}{2}\varepsilon^T\Omega\varepsilon + \zeta^T\varepsilon\}, s.t. \begin{cases} A \times \varepsilon \leq b \\ Aeq \times \varepsilon = beq \\ lb \leq \varepsilon \leq ub \end{cases}, \quad (8)$$

where $\Omega$, $A$, and $Aeq$ are matrices, $\zeta$, $b$, $beq$, $lb$, and $ub$ are vectors, and $\varepsilon$ denotes the change in a core's frequency. The key point in MPC design is to transform the problem defined in (4)-(7) to the standard quadratic programming format in (8). Based on $\varepsilon$ and the core's frequency inside $SW_k$, the core's frequency at $SW_{k+1}$ can be directly calculated. We omit the details in the transformation to save space. After the transformation, we can directly solve the quadratic programming problem using a standard solver, e.g., *quadprog* tool in Matlab.

## 4.3 Sampling window control

The proposed GUC aims to reduce the temperature of the entire multicore chip and to balance the temperature differences among cores, but may introduce thermal cycles. Since a core's frequency and voltage can change from one sampling window to another, $L_{sw}$ directly impacts thermal cycles. It is difficult to precisely model how the sampling window length affects system reliability, and the complicated reliability model makes finding the best sampling period too time consuming for online adoption. Hence, to judiciously balance the impact of temperature against that of thermal cycles on system MTTF, we design an efficient online heuristic based on binary search to adjust $L_{sw}$ at runtime.

Since a core's temperature is treated as constant inside each sampling window, the aging effect of thermal cycles is ignored if we only consider a single sampling window. In SWC, we use the concept of profiling window to adjust $L_{sw}$. A profiling window is composed of $s$ sampling windows with the same length and whose $s$ temperature points make up a temperature profile. At the end of a profiling window, we establish the system MTTF from the temperature profile using a reliability modelling tool, e.g., [10]. In our experimental evaluations, we find that $s$ = 100 achieves an accurate system MTTF with an acceptable overhead in MTTF calculation. We store the history of the system MTTF for the various lengths of the sampling window and rely on binary search to find the most appropriate $L_{sw}$ for the sampling windows in the next profiling window.

**Algorithm 1:** SWC

```
1:  initialize $Range_i$, $Range_j$, and $L^c_{sw}$, set i=j=1
2:  while True then
3:      if at the end of $SW_i$ then
4:          measure temperature $T_k(i)$ for each core
5:          $i \leftarrow i+1$
6:          if $i = s$ then
7:              $TP(j) \leftarrow [T_i(1), \cdots, T_m(i)]$
8:              calculate $MTTF_j$
9:              if $j = 1$ then
10:                 $L^c_{sw} \leftarrow L^{ub}_{sw}$
11:             else
12:                 $L^c_{sw} \leftarrow (Range_i + Range_j)/2$
13:                 if $MTTF_j \geq MTTF_{j-1}$ then
14:                     $Range_i \leftarrow (Range_i + Range_j)/2$
15:                 else
16:                     $Range_j \leftarrow (Range_i + Range_j)/2$
17:             $i \leftarrow 1$, $j \leftarrow j+1$
18: end while
```

To find $L_{sw}$, let [$Range_i$, $Range_j$] denote the binary search range. We initially set $Range_i$ to 1 second to keep the overhead due to RUC under 1% of the sampling window length (based on an Intel Core i5 2.6GHz and 2GB memory). To determine an initial value for $Range_j$, we observe that since GUC expects a constant temperature in a given sampling window, so the sampling window length must be smaller than some constant $C_{HW}$, which depends on runtime environment and hardware platform, and is typically on the order of 10 seconds. Moreover, since different hyperperiods ($HP$s) of a given periodic task set have the same workload, a core's frequency remains the same across different $HP$s if $L_{sw} = HP$ and utilization set point is fixed. Considering that a larger $L_{sw}$ leads to a higher temperature, the appropriate value for $L_{sw}$ should be less than or equal to $HP$. As a result, we initialize $Range_j$ to min{$HP$, $C_{HW}$}. The pseudo code for SWC is given in Alg.1, where $L^c_{sw}$ is $L_{sw}$ in the current sampling window. The search area, [$Range_i$, $Range_j$], is updated in each iteration until the most appropriate value for $L_{sw}$ is found.

## 5. EVALUATION

In this section, we present simulation results to evaluate the effectiveness of RUC. An event-driven simulator was implemented to simulate task execution. Tasks are scheduled using EDF [11] and executed to completion even if they have missed their deadlines. Each benchmark is composed of 50 randomly generated task sets. In each benchmark, the utilization level is kept constant. The utilization levels of the task set considered are 50,60,...,90%. It is not necessary to evaluate the effectiveness of RUC under low utilization levels since the lowest frequency level would allow all the deadlines to be met while minimizing the core temperatures. To model different task behaviors, we studied two cases: fixed and variable task execution times. For the latter case, actual execution times of the jobs are obtained by multiplying the specified execution times by a random value in the range between [0.9, 1.1].

The hardware platform under consideration consists of 4 homogeneous cores and each of them is constructed based on an ALPHA 21264 microprocessor [2]. The processor parameters and constants are listed in TABLE I. The initial temperature is assumed to be 330°F (56.85°C) and the ambient

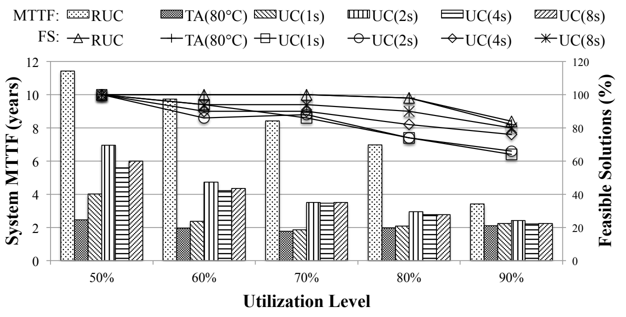

**Figure 2: MTTF and FS values obtained by RUC, UC, and TA when tasks' execution times are fixed.**

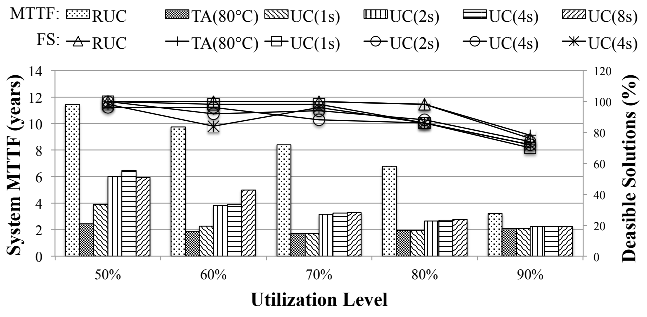

**Figure 3: MTTF and FS values obtained by RUC, UC, and TA when tasks' execution times vary dynamically.**

temperature is 45°C. The runtime temperature is calculated using Hotspot [13].

**Table 1: Processor Parameters and Constants [2]**

| Voltage (V) | Frequency (GHz) | $\alpha_1$ | $\alpha_2$ | $\alpha_3$ |
|---|---|---|---|---|
| 0.7 | 1.00 | 1.422 | 0.1903 | 15.0 |
| 0.8 | 1.25 | 2.916 | 0.2111 | 15.0 |
| 0.9 | 1.5 | 5.41 | 0.234 | 15.0 |
| 1.0 | 1.75 | 9.702 | 0.2592 | 15.0 |
| 1.1 | 2.00 | 17.22 | 0.2867 | 15.0 |

We compare the performance of RUC against two representative controller based mechanisms: utilization control (UC) and temperature-aware (TA). Similar to the existing work in [12], UC controls a core's frequency to force the utilization to converge to a chosen set point, but it has a fixed length of sampling window. In our simulations, $L_{sw}$ is 1 second (UC(1s)), 2 seconds (UC(2s)), 4 seconds (UC(4s)) and 8 seconds (UC(8s)). As for TA [4], each core's temperature is controlled to converge to its chosen set point. The temperature set point is 80°C, which is below the temperature threshold for hardware throttling [4]. We conduct the comparison between RUC and TA to show that temperature reduction alone is suboptimal in maximizing system MTTF.

We compare RUC, UC and TA in terms of MTTF and real-time performance. The real-time performance is quantified as the percentage of feasible solutions (FS), which is the ratio of the number of task sets satisfying their real-time requirements over the total number of task sets. For UC and RUC, the utilization set point is 90% as in [12]. When tasks execution times are fixed (see Fig.2), the average MTTFs due to RUC for different benchmarks are 11.42, 9.73, 8.43, 6.98, and 3.42 years, respectively. RUC increases MTTF values by up to nearly 364% and at least 52%. In addition, using RUC always results in the highest FS compared to the other mechanisms in all the benchmarks. These results show that by controlling utilization and adjusting the sampling window length, more real-time task sets can be feasibly scheduled while maximizing system lifetime. RUC exhibits a similar improvement when tasks' execution times vary at runtime (see Fig.3). Its average MTTFs are 11.43, 9.74, 8.40, 6.78, and 3.22 years for the five benchmarks. RUC improves system MTTF by 43% - 369% while maintaining FS at a high level. We can conclude that for a variety of different runtime environments, RUC is effective in improving system lifetime and increasing the schedulability of real-time tasks.

## 6. CONCLUSIONS

We proposed a reliability-aware utilization control framework to maximize the lifetime of multicore systems under soft real-time constraints. Our mechanism jointly minimizes core temperatures, the temperature differences among cores, and thermal cycling. It uses a model predictive controller to control the utilization of real-time tasks by adjusting a core's frequency/voltage within each sampling window. We also introduced a heuristic to dynamically determine the length of a sampling window. Simulation results reveal that our approach is indeed effective in increasing the lifetime of soft real-time systems. As future work, we plan to extend our mechanism to heterogeneous multicore and hard real-time systems.

## References

[1] J. Srinivasan, et al., "The impact of technology scaling on lifetime reliability," in *Proc. Dependable Systems and Networks*, Jun. 2004, pp. 177–186.

[2] G. Quan and V. Chaturvedi, "Feasibility analysis for temperature-constraint hard real-time periodic tasks," *IEEE Trans. Industrial Informatics*, vol. 6, no. 3, pp. 329–339, Aug. 2012.

[3] T. Chantem, R. Dick, and X. Hu, "Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs," *IEEE Trans. VLSI Systems*, vol. 19, no. 10, pp. 1884–1897, Oct. 2011.

[4] Y. Fu, et al., "Feedback thermal control of real-time systems on multicore processors," in *Proc. of int. conf. on Embedded software*, Oct. 2012, pp. 113–122.

[5] JEDEC Solid State Technology Association, "Failure mechanisms and models for semiconductor devices," *JEDEC Publication*, 2003.

[6] A. Coskun, et al., "Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors," in *Proc. Conf. Measurement and Modeling of Computer System*, Jun. 2009, pp. 169–180.

[7] A. Hartman, D. Thomas, and B. Meyer, "A case for lifetime-aware task mapping in embedded chip multiprocessors," in *Proc. Conf. Hardware/Software Codesign and System Synthesis*, Oct. 2010, pp. 145–154.

[8] A. Hartman and D. Thomas, "Lifetime improvement through runtime wear-based task mapping," in *Proc. Hardware/Software Codesign and System Synthesis*, Oct. 2012, pp. 13–22.

[9] T. Chantem, et al., "Enhancing multicore reliability through wear compensation in online assignment and scheduling," in *Proc. Design, Automation and Test in Europe*, Mar. 2013, pp. 1373–1378.

[10] Y. Xiang, et al., "System-level reliability modeling for MPSoCs," in *Proc. on Hardware/Software Codesign and System Synthesis*, Oct. 2010, pp. 297–306.

[11] C. Liu and J. Layland, "Scheduling algorithm for multiprogramming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973.

[12] C. Lu, X. Wang, and X. Koutsoukos, "Feedback utilization control in distributed real-time systems with end-to-end tasks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 16, no. 6, pp. 550–561, Jun. 2005.

[13] K. Skadron, et al., "Temperature-aware microarchitecture: Modeling and implementation," *ACM Trans. on Architecture and Code Optimization*, vol. 1, no. 1, pp. 94–125, Mar. 2004.