Enhancing Multicore Reliability through Wear Compensation in Online Assignment and Scheduling

Thidapat Chantem
Department of ECE
Utah State University
Logan, UT 84322
tam.chantem@usu.edu

Yun Xiang
Department of EECS
University of Michigan
Ann Arbor, MI 48109
xiangyun@umich.edu

X. Sharon Hu
Department of CSE
University of Notre Dame
Notre Dame, IN 46556
shu@nd.edu

Robert P. Dick
Department of EECS
University of Michigan
Ann Arbor, MI 48109
dickrp@umich.edu

***Abstract*—System reliability is a crucial concern especially in multicore systems which tend to have high power density and hence temperature. Existing reliability-aware methods are either slow and non-adaptive (offline techniques) or do not use task assignment and scheduling to compensate for uneven core wear states (online techniques). In this article, we present a dynamically-activated task assignment and scheduling algorithm based on theoretical results that explicitly optimizes system lifetime. We also propose a data distillation method that dramatically reduces the size of the thermal profiles to make full system reliability analysis viable online. Simulation results show that our algorithm results in between 27–291% improvement to system lifetime compared to existing techniques for four-core systems.**

I. Introduction & Contributions

While multicore systems offer superior performance and power efficiency, reliability issues have become a major concern. Shrinking device sizes and increasing transistor counts result in high power density and hence temperature. System reliability is a strong function of temperature; a 10–15 °C difference in operating temperature can result in a 2× difference in the lifespan of a device [1]. For many applications, maximizing system lifetime is an important and challenging goal in order to avoid the cost of replacing an entire system and maintain quality of service (QoS) such as user's satisfaction.

According to Hartman et al., temperature-aware approaches (e.g., [2]–[9]) are not sufficient in maximizing system lifetime [10]. There are a few research papers on reliability-aware multicore systems (e.g., [10]–[12]). However, since finding a task assignment and scheduling solution that optimizes the system mean time to failure (MTTF) usually involves some type of search algorithm, existing approaches usually only provides offline solution that cannot adapt to dynamic changes in application requirements. Coskun et al. presented a collection of online techniques to reduce a job's impact on system reliability [12]. However, these techniques do not directly consider the wear state of a core and do not compensate for wear imbalance, which reduces system MTTF. The work by Paterna et al. exploited core activity duty cycling to regulate the percentage of idle time on cores that are more likely to fail [13] but did not specifically account for thermal cycling. Hartman and Thomas [14] relied on the existence of wear sensors to remap tasks online. Unfortunately, wear sensors [15], [16] are not yet widely available and cannot detect all common IC failure mechanisms. In contrast to existing work, we make the following main contributions.

1) We analytically determine the thermal profile that maximizes system MTTF for any given workload. Our analysis allows for a trade-off between two factors that significantly impact system reliability: temperature and thermal cycling.
2) We present an online task assignment and scheduling algorithm to maximize system lifetime.
3) To reduce runtime overhead, we propose a method to adaptively adjust the activation frequency of our online algorithm.
4) To make full system reliability analysis viable online, we propose a high-fidelity data reduction method to compress potentially large temperature traces using temperature and frequency bins.

II. System Model and Assumptions

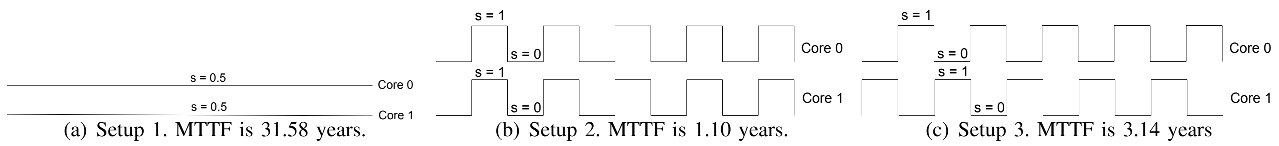
The system consists of $|M|$ heterogeneous cores, $m_0, m_1, \cdots, m_{|M|}$. Each core is equipped with a temperature sensor that continuously outputs the core's temperature. Any reading inaccuracy due to temperature sensor time lag can be ignored since it is extremely unlikely for a core's temperature to change drastically within that time lag. The system operating temperature is between 45 °C and 100 °C and no processor throttling takes place in this temperature range.

The *system workload* consists of a number of non-preemptive, periodic, and independent tasks. The workload may change over time due to changes in user inputs or system state. When it is executing on core m_j, task τ_i has an execution time $C_{i,j}$. The utilization of τ_i on m_j is the percentage of time τ_i requires from m_j. Task τ_i is also characterized by $\overline{p}_{i,j}$, which is its average power consumption when executing on m_j. Since we consider a heterogeneous system, we define $\overline{p}_i$ to be the average power consumption of τ_i when executing on some reference processor. This makes it possible to rank tasks by power consumption. No migration is allowed once a task starts executing on a core.

There are several device lifetime failure mechanisms that are presently dominant for ICs: electromigration (EM) [17], [18], time-dependent dielectric breakdown (TDDB) [17], [19], stress migration (SM) [17], and thermal cycling (TC) [20]. Please refer to our technical report [21] for details.

To maximize system MTTF, we start by noting that in general, the lower the temperature, the better the reliability. This observation has been made by several researchers, e.g.,

This work was supported in part by Utah State University, in part by a research contract from the Sandia National Laboratories, and in part by the NSF under award CCF-0964763.
978-3-9815370-0-0/DATE13/©2013 EDAA



(a) Setup 1. MTTF is 31.58 years. (b) Setup 2. MTTF is 1.10 years. (c) Setup 3. MTTF is 3.14 years

Fig. 1: Impact of thermal cycling on system MTTF.

[10]–[12]. For EM, TDDB, and SM, the failure rates are exponential functions of temperature. For TC, however, the amplitudes and frequencies of the thermal cycles matter as much as the maximum temperatures of those cycles. While some existing work has highlighted the importance of TC and proposed techniques to reduce the frequencies and amplitudes of thermal cycles [12], a general solution to achieve the conflicting goal of reducing temperature and reducing the amplitudes and frequencies of thermal cycles is still needed.

III. Ideal Homogeneous Cores

We first focus on the ideal setup where the system is homogeneous. Details on our reliability model as well as proofs to lemmas and theorems can be found in our technical report [21].

Theorem 1. *Assume a 2^n-core system which consists of homogeneous cores with identical lateral and vertical thermal and electrical characteristics and identical initial wear state at time t. Further assume that the power distribution is fixed during the time interval $[t, t + \Delta t]$ for each core. Then at time $t + \Delta t$ the system MTTF is maximized if and only if the power is distributed evenly among the cores during $[t, t + \Delta t]$.*

It follows that to maximize system MTTF, perfect spatial and temporal load balancing must be achieved. In other words, the maximum core wear state among all the cores must be minimized. We use an example to demonstrate the significant impact of load balancing on system lifetime. (For this example and the rest of the paper, system MTTF values are obtained using a system-level reliability modeling tool [22].) Consider two ideal homogeneous cores with a maximum power consumption of 65 W. There are ten tasks with a utilization of 0.1 each. The system MTTF is only 3.27 years if all the tasks are executed on only one core. On the other hand, if the workload is evenly distributed (i.e., five tasks per core), the system MTTF is 31.6 years, which is almost a $10\times$ improvement.

Due to task execution granularity and performance constraints such as response times, achieving absolute spatial and temporal load balancing is rarely possible. The goal, then, is to understand which types of load unbalances have the smallest impacts on system reliability.

IV. Non-Ideal Cores

We discuss lifetime optimization on heterogeneous systems and the trade-offs between temperature and thermal cycles.

A. Heterogeneous Core and Cores with Uneven Wear States

We start with some general observations pertaining to any multicore system. For a system that initially consists of perfectly homogeneous cores, the cores will eventually have different wear states unless perfect spatial and temporal load balancing is maintained at all times. One way to slow down the wear process is to use *wear compensation*, which is the process of assigning loads to cores in such a way that the maximum difference among all core wear states is minimized. For a system that initially consists of perfectly homogeneous cores and for which perfect spatial and temporal load balancing cannot be achieved, wear compensation helps to maximize system MTTF, as demonstrated by the following example.

Consider again the multicore system in Section III. Assume that the total system utilization is 1.6, which means that both cores are needed by the system. Ideally, each core should execute half of the workload. However, workload may not be perfectly divisible. Assume, for example, that the task set consists of 5 tasks $\tau_1, \tau_2, \ldots, \tau_5$, where the utilization of τ_1 and τ_2 is 0.5 each and the utilization of τ_3, τ_4, and τ_5 is 0.2 each. If τ_1, τ_3, and τ_4 are assigned to m_0 and τ_2 and τ_5 on m_1, the system MTTF is 1.57 years. If wear compensation is applied by swapping the loads on the cores after half of the time duration, the system MTTF increases to 1.95 years, which is a 24% improvement.

We can also apply the concept of wear compensation to heterogeneous cores. That is, for such a system, maximizing its MTTF is equivalent to maximizing the MTTF of a homogeneous system where cores have unequal wear.

B. Trade-Offs Between Temperature and Thermal Cycles

In contrast to wear due to EM, TDDB, and SM, which primarily depend on temperature, wear due to TC also depends on the amplitudes and frequencies of the thermal cycles. We use a simple example to illustrate the impact of thermal cycles on system lifetime. Let us reconsider the system and task set from earlier. In addition, consider the three task execution patterns shown in Fig. 1. In the first two setups, the system is spatially balanced. However, the thermal profile from Setup 2 contains large and frequent thermal cycles, which cause system MTTF to be reduced by more than $30\times$. On the other hand, a thermal profile that is neither spatially nor temporally balanced but with smaller thermal cycles (Setup 3) lead to better system MTTF.

For the rest of this section, we describe how to determine the most desirable thermal profile for the practical scenarios where thermal cycles cannot be avoided. Essentially, for some fixed workload, we wish to determine the execution (and hence power) profile that maximizes system reliability. In order to compare any two thermal profiles $\mathbf{T}_1$ and $\mathbf{T}_2$, we observe that there are three aspects of thermal cycling, which lead to eight different cases (or four unique cases), as shown in Tab. I. We now present each unique case.

Lemma 1. *Given two thermal profiles $\mathbf{T}_1$ and $\mathbf{T}_2$. Let ΔT_1 and ΔT_2 denote the maximum amplitudes of all thermal cycles in $\mathbf{T}_1$ and $\mathbf{T}_2$, respectively, C_1 and C_2 be the numbers of thermal cycles in $\mathbf{T}_1$ and $\mathbf{T}_2$, respectively, and $T_{\max 1}$*

TABLE I: A comparison between any two thermal profiles will result in one of these four cases.

Case	Amplitudes	Frequencies	Max. temps.	Reliability
1	$\Delta T_1 \geq \Delta T_2$	$C_1 \geq C_2$	$T_{\max 1} \geq T_{\max 2}$	$MTTF_1 \leq MTTF_2$
2	$\Delta T_1 \geq \Delta T_2$	$C_1 \geq C_2$	$T_{\max 1} \leq T_{\max 2}$	$MTTF_1 \leq MTTF_2$ if $\frac{C_1}{C_2} \geq 9.56$ or $\frac{\Delta T_1}{\Delta T_2} \geq 3.09$
3	$\Delta T_1 \geq \Delta T_2$	$C_1 \leq C_2$	$T_{\max 1} \geq T_{\max 2}$	$MTTF_1 \leq MTTF_2$ if $\frac{C_2}{C_1} \leq \left(\frac{\Delta T_1}{\Delta T_2}\right)^2$
4	$\Delta T_1 \geq \Delta T_2$	$C_1 \leq C_2$	$T_{\max 1} \leq T_{\max 2}$	$MTTF_1 \leq MTTF_2$ if $\left(\frac{\Delta T_1}{\Delta T_2}\right)^2 \geq 9.56\frac{C_2}{C_1}$ or $MTTF_1 \geq MTTF_2$ if $\frac{C_2}{C_1} \geq \left(\frac{\Delta T_1}{\Delta T_2}\right)^2$

and $T_{\max 2}$ be the maximum temperatures in $\mathbf{T}_1$ and $\mathbf{T}_2$, respectively. Assume that the system operating temperature is between 45 °C and 100 °C. Further assume that the number of cycles to failure due to $\mathbf{T}_i$ is $N_i = A \cdot \Delta T_i^{-b} e^{\frac{E_a}{\kappa T_{\max i}}}$, where A is some constant, $E_a = 0.42$, $\kappa = 8.62 \times 10^{-5}$, and $b = 2$. If $\Delta T_1 \geq \Delta T_2$, $C_1 \geq C_2$, and $T_{\max 1} \geq T_{\max 2}$, then $MTTF_1 \leq MTTF_2$.

Lemma 2. *Given two thermal profiles $\mathbf{T}_1$ and $\mathbf{T}_2$, the system operating temperature range, and the number of cycles to failure as described in Lemma 1. If $\Delta T_1 \geq \Delta T_2$, $C_1 \geq C_2$, and $T_{\max 1} \leq T_{\max 2}$, then $MTTF_1 \leq MTTF_2$ if $\frac{C_1}{C_2} \geq 9.56$ or $\frac{\Delta T_1}{\Delta T_2} \geq 3.09$.*

Lemma 3. *Given two thermal profiles $\mathbf{T}_1$ and $\mathbf{T}_2$, the system operating temperature range, and the number of cycles to failure as described in Lemma 1. If $\Delta T_1 \geq \Delta T_2$, $C_1 \leq C_2$, and $T_{\max 1} \geq T_{\max 2}$, then $MTTF_1 \leq MTTF_2$ if $\frac{C_2}{C_1} \leq \left(\frac{\Delta T_1}{\Delta T_2}\right)^2$.*

Lemma 4. *Given two thermal profiles $\mathbf{T}_1$ and $\mathbf{T}_2$, the system operating temperature range, and the number of cycles to failure as described in Lemma 1. If $\Delta T_1 \geq \Delta T_2$, $C_1 \leq C_2$, and $T_{\max 1} \leq T_{\max 2}$, then $MTTF_1 \leq MTTF_2$ if $\left(\frac{\Delta T_1}{\Delta T_2}\right)^2 \geq 9.56\frac{C_2}{C_1}$ or $MTTF_1 \geq MTTF_2$ if $\frac{C_2}{C_1} \geq \left(\frac{\Delta T_1}{\Delta T_2}\right)^2$.*

A summary of our findings is provided in Tab. I. Numerical values 9.56 and 3.09 depend on the system operating temperature range.

V. Online Algorithm

We now present our online reliability-aware task assignment and scheduling algorithm for multicore systems.

A. Overview & Formal Problem Statement

One way to achieve perfect spatial and temporal load balancing is to perform constant task migration. Unfortunately, such a solution is not practical due to migration overhead. The solution also assumes a continuous stream of infinitesimal tasks that can be distributed in a perfectly balanced manner, which is unrealistic. In our approach, we adopt an online algorithm to solve the reliability problem in multicore systems. We focus on periodically assigning and scheduling tasks to compensate for existing spatial and temporal load imbalances without unnecessarily increasing the schedule length. Specifically, we aim to solve the following problem.

Problem 1: Given a system of heterogeneous cores with some initial wear states, determine online task to core assignment and task scheduling such that the schedule length is minimized and system MTTF is maximized.

B. Reliability-Aware Task Assignment and Scheduling

The main idea of our algorithm is to balance core wear states during the task assignment phase without putting unnecessarily loads on any given core so as to slow down the wear process. Then, during the task scheduling phase, we use our prior knowledge of the desirable thermal profile to schedule tasks on a given core. Our algorithm is described in Alg. 1. The algorithm takes as inputs a set of cores M, a set of tasks Γ, and a system parameter T_{cutoff}, which is used to decide whether a core is considered hot or cool. We start by estimating the wear state of each core using some system-level reliability modeling tool (Line 1). Such a tool returns a structure r_i for core m_i, which contains wear state due to EM ($r_i.EM$), TDDB ($r_i.TDDB$), SM ($r_i.SM$), TC ($r_i.TC$), as well as the overall wear state of core m_i ($r_i.totalWear$). To slow down the wear process on the cores by as much as possible, we adopt a largest-task first (LTF)-based algorithm [23] to assign tasks to cores. The LTF-based algorithm, which attempts to spatially balance load, processes tasks in a non-increasing order of energy consumption and assign them to the core with the least total energy consumption (Lines 2–6). Once a task is assigned to a core, the core's total energy consumption is updated. Since the power consumption and execution time of each task on a core is known (Section II), the task's energy consumption can be easily calculated. Also, while we do not consider dynamic voltage and frequency scaling (DVFS) here, existing DVFS techniques can be readily incorporated.

To make scheduling decisions, we start by comparing the wear states of the cores in the system to those of the least worn core. Specifically, we first determine, for the least worn core, the average core wear state due to EM, SM, and TDDB (Line 7), as well as the core wear state due to TC (Line 8). Recall that while wear due to EM, SM, and TDDB primarily depend on temperature, wear due to TC also depends on the amplitudes and frequencies of the thermal cycles. In the following for-loop (Lines 9–14), Alg. 1 classifies each core in the system according to its wear characteristics (Lines 11–14).

Next, each core is classified as either hot (Line 16) or cool (Line 28) by comparing the core's current temperature from the appropriate temperature sensor to T_{cutoff}. If a core is only worn due to TC, cycles should be avoided (Lines 17–18) and tasks are ordered in a non-increasing (non-decreasing, resp.) order of average power consumption for a hot (cool, resp.) core. Average power is a good estimation of a task's impact on a core's temperature since temperature is a function of power. That is, a hot (cool, resp.) core will execute tasks that are expected to raise (lower, resp.) the core's temperature first to ensure a smoother thermal profile, i.e., one with fewer cycles.

Let $\overline{\mathbf{p}} = \{\overline{p}_i\}$, $i = 1, \ldots, |\Gamma|$. If a core is worn due to high temperature (Lines 19–20), tasks are scheduled in the following order for the hot core: task with the lowest value in $\overline{\mathbf{p}}$, task with the highest value in $\overline{\mathbf{p}}$, task with the second

Algorithm 1 Online_Rel_Aware_Algorithm(M, Γ, T_{cutoff})

```
1: r ← sys_rel($M$) // Compute wear state of each core
2: sort $M$ in a non-decreasing order of $r_i.totalWear$, $i = 1, \ldots, |M|$
3: sort $\Gamma$ in a non-increasing order of energy consumption $E_i = \overline{p}_i \cdot C_i$, $i = 1, \ldots, |\Gamma|$
4: $M' \leftarrow$ LTF_Energy($\Gamma$, $M$) // $M'$ is sorted in a non-increasing order of total energy of assigned tasks
5: for $i = 1, \ldots, |M|$ do // Least worn cores receive largest load
6:   assign tasks on $m'_i$ to $m_i$, $m'_i \in M'$, $m_i \in M$
7: $w^*_T \leftarrow \frac{r_0.EM+r_0.SM+r_0.TDDB}{3}$ // $m_0$: least worn core
8: $w^*_{TC} \leftarrow r_0.TC$
9: for each $m_i \in M$ do // Compare wear states of each core to those of the least worn core to identify weaknesses
10:   $stateT_i \leftarrow false$, $stateTC_i \leftarrow false$
11:   if $\frac{r_i.EM+r_i.SM+r_i.TDDB}{3} > w^*_T$ then
12:     $stateT_i \leftarrow true$
13:   if $r_i.TC > w^*_{TC}$ then
14:     $stateTC_i \leftarrow true$
15: for each $m_i \in M$ do // Order tasks on this core to optimize the thermal profile according to the weaknesses of the core
16:   if $Tcurr_i > T_{cutoff}$ then // Core temperature from sensor exceeds the cut-off temperature and is considered hot
17:     if $stateT_i = false$ and $stateTC_i = true$ then // Case A: avoid cycles
18:       schedule tasks in a non-increasing order of $\overline{p}_i$, $i = 1, \ldots, |\Gamma|$
19:     else if $stateT_i = true$ and $stateTC_i = false$ then // Case B: avoid high temperature
20:       schedule tasks in the following order: $\text{argmax}_{i=1,\ldots,|\Gamma|}\overline{p}_i$, $\text{argmin}_{i=1,\ldots,|\Gamma|}\overline{p}_i$, ..., $i = 1, \ldots, |\Gamma|$
21:     else
22:       $result$ ← compare Cases A and B according to Tab. I
23:       if $result = inconclusive$ then
24:         if $r_i.TC > \frac{r_i.EM+r_i.SMr_i.TDDB}{3}$ then
25:           apply case A
26:         else
27:           apply case B
28:   else // Core temperature is below the cut-off temperature and is considered cool
29:     // Similar to the case where the core is considered hot (Lines 17–27)
```

lowest value in $\overline{\mathbf{p}}$, and so on. In this way, a thermal profile with more thermal cycles but lower temperature is achieved. If the core is cool, tasks are scheduled in the same manner except that we start with the task with the highest value in $\overline{\mathbf{p}}$. Finally, if both high temperature and large and frequent thermal cycles should be avoided, we use the conditions in Tab. I to determine the best course of action (Lines 21–22). However, since there may be cases that fall outside the conditions in Tab. I, a definite answer cannot always be obtained. In such a case, task scheduling decisions are based on whether the core under consideration is more worn due to high temperature or cycles (Lines 23–27). The time complexity of our algorithm is $O(|M| \cdot |\Gamma| + |M| \cdot TA + R)$, where $O(TA)$ and $O(R)$ are the run-time complexity of a thermal analysis tool and a system-level reliability modeling tool, respectively.

Algorithm 2 Dyn_Calibration_Interval(M, $e_{\max}$, e_{old})

```
1: $e \leftarrow \max_{\forall m_i, m_j \in M} |r_i.totalWear - r_j.totalWear|$
2: if $e \geq e_{\max}$ or $e \geq e_{old}$ then
3:   activate Algorithm 1
4: return $e$
```

VI. Practical Considerations

There are still unanswered questions. First, we need to determine how often to invoke Alg. 1. Second, most system-level reliability modeling tools tend to be slow because they use Monte Carlo simulation to find system MTTF and the size of the thermal trace needed can be very large depending on the invocation period of the algorithm. Third, we have not considered the overhead of using our algorithm online.

A. Algorithm Activation Frequency

Alg. 1 should only be activated infrequently to reduce runtime overhead. In addition, a periodic invocation of Alg. 1 may not be desirable since a core's wear state is a nonlinear function that depends on current wear state. Depending on the current wear state, it may take anywhere from a few days to several months to increase system wear state by 10%.

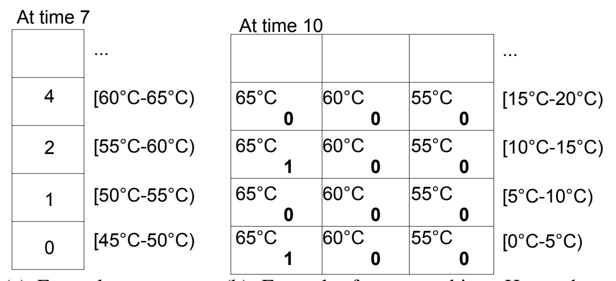
We propose to dynamically calibrate the activation frequency of Alg. 1 by leveraging a control-based approach, as shown in Alg. 2. There are two situations that will invoke Alg. 1: (i) if the maximum difference in core wear state is larger than the maximum tolerable difference $e_{\max}$, which can be obtained via profiling, and (ii), if the maximum difference among the wear states of the cores are increasing over time.

B. Reliability Data Reduction Method

While a complete system-level reliability analysis tool allows us to accurately determine the wear state of each core, it takes as an input a thermal trace, which can be in gigabytes for a one-day run. To reduce the size of the data that must be kept, we propose using bins to collect temperature values.

For failure mechanisms such as EM, TDDB, and SM, which depend on actual core temperature, we use a number of *temperature* bins for each core. Each bin is labeled with a temperature range and contains the total duration a core spends at that temperature range since the last algorithm activation time, as shown in Fig. 2(a). Since the ordering of the temperature points does not matter for EM, TDDB, and SM analyses, the temperature bins, each of which contains a single float value, are a viable solution.

For thermal cycling, we observe that TC depends not only on the frequencies of the cycles, but also on the maximum temperatures during the cycles and their amplitudes. Hence, collecting only the temperature data or the frequency data is insufficient and can lead to large inaccuracies. For this reason, we leverage the structure of each *frequency* bin to capture the essential characteristics of each thermal cycle. Each bin is labeled with a thermal cycle range and contains a number of sub-bins. Each sub-bin, which is tagged with a maximum temperature range, keeps track of the number of times (i.e., frequency) a core experiences cycles in that range with the


At time 7
...
4 [60°C-65°C)
2 [55°C-60°C)
1 [50°C-55°C)
0 [45°C-50°C)
At time 10
...
65°C 0 60°C 0 55°C 0 [15°C-20°C)
65°C 1 60°C 0 55°C 0 [10°C-15°C)
65°C 0 60°C 0 55°C 0 [5°C-10°C)
65°C 1 60°C 0 55°C 0 [0°C-5°C)

(a) Example temperature bins.

(b) Example frequency bins. Here, there are three sub-bins for each frequency bin. For example, the leftmost sub-bin of the $[0°\text{C}, 5°\text{C})$ bin indicates that one cycle of less than $5°\text{C}$ occurred at a maximum temperature between $60°\text{C}$ and $65°\text{C}$.

Fig. 2: Bin example.

maximum temperature falling inside the tagged maximum temperature range. Fig. 2(b) shows some example frequency bins. Note that thermal cycles can be detected online using the one-pass rainflow counting algorithm [24].

The number of bins is chosen *a priori*. Offline simulations can be performed to select the appropriate number of temperature and frequency bins. For temperature bins, a set of simulations involving 10,000 randomly generated datapoints showed that having each bin span about 0.5 °C (e.g., a bin for 80 °C–80.5 °C) is sufficient to maintain over 99% accuracy in core wear states. Roughly, this translates to 110 bins for our system. For frequency bins, about 500 bins are needed, along with 500 sub-bins, for a total of 250,000 bins to maintain 96% accuracy. While a quarter of a million bins may seem substantial, our data distillation method reduces storage space from several gigabytes to just a few kilobytes (for temperature bins) and a few megabytes (for frequency bins). Last but not least, since the run time of the reliability modeling tool also depends on the size of the thermal trace, we have effectively reduced the overhead associated with using the tool online. In summary, our proposed algorithm is viable for online use since it only needs to be activated every few days at the maximum and since the size of the thermal trace is substantially reduced.

VII. Simulation Results

We compare the performance of our reliability-aware task assignment and scheduling algorithm to the following set of representative algorithms that have been proposed in the past.

Ideal algorithm (IA) where perfect spatial and temporal load balancing are assumed. The resultant system MTTF is the theoretical minimum and is provided for comparison only.

Random algorithm (RA) where task assignment and scheduling is performed at random.

Energy-aware algorithm (EA) where loads are balanced spatially but not temporally. Tasks are ordered in a non-decreasing order of energy consumption and assigned to the core with the minimum aggregated energy consumption.

Temperature-aware algorithm (TA) where temperature is reduced by alternately executing hot and cool tasks. This algorithm is loosely based on an algorithm proposed by Huang et al. [25]. Tasks are ordered in a non-decreasing order of average power consumption and assigned to the core with the minimum aggregated average power consumption.

We omit comparisons with offline techniques (due to their inability to adapt) and online techniques that involve frequent task migration since such techniques may incur large data transfer overhead and are likely to change cache behavior.

Ten sets of benchmarks with 100 tasks and ten sets of benchmarks with 1000 tasks were randomly generated. Task execution times were generated using an exponential distribution ($\lambda = 1$). Task power consumption segments follow a uniform distribution. We assume that all tasks arrive at the same time to model the worst-case workload. Our systems consist of four or nine homogeneous cores. Each core is based on the Alpha 21264 processor, with a maximum power consumption of 120 W @ 4 GHz [6]. For a given system, initial wear state can either be even or uneven. The initial temperature at the start of the simulation is 60 °C and $T_{cutoff} = 72.5$ °C. After a benchmark completes, the resultant thermal profile from HotSpot 5.0 [26] is assumed to repeat. A system-level reliability modeling tool [22] is used to obtain reliability data.

The normalized system MTTFs for the benchmarks with 100 tasks are shown in Figs. 3(a) and 3(b) for four-core systems with initially even and uneven core wear states, respectively. It is not surprising that IA produces solutions that always maximizes system MTTF. Our reliability-aware algorithm outperforms RA, EA, and TA by 39%, 63%, and 103% on average, respectively, for the system with initially even core wear states, and 27%, 58%, and 97% on average, respectively, for the system with initially uneven core wear states. TA, which may be expected to perform better, is the worst since it creates thermal cycles without regards to core wear states.

For benchmarks with 1000 tasks, the normalized system MTTFs are as shown in Figs. 3(c) and 3(d) for four-core systems with initially even and uneven core wear states, respectively. Again, our reliability-aware algorithm outperforms RA, EA, and TA by 149%, 162%, and 291%, respectively for the system with initially even core wear states. For the system with initially uneven core wear states, the improvements are 142%, 157%, and 287%, respectively. It is clear that as the number of tasks in the system increases, existing methods do not come close to maximizing system reliability.

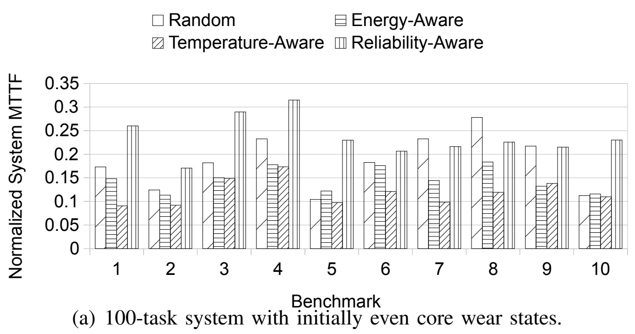
Finally, for nine-core systems, the results are more modest. For benchmarks with 100 and 1000 tasks, the improvements are between -4% to 23% and 8% to 128%, respectively. RA sometimes outperforms our algorithm for smaller benchmarks, though the average improvements of our algorithm over RA for larger benchmarks are between 8% to 15% [21].

VIII. Conclusions & Future Work

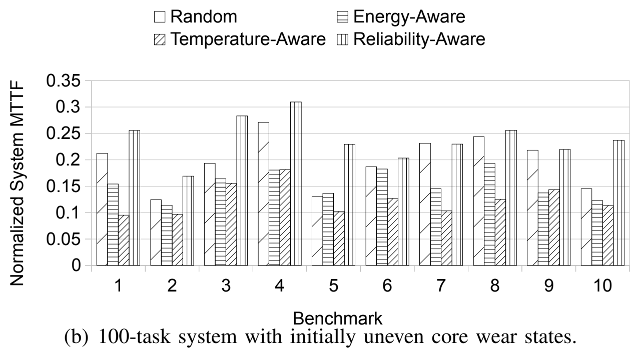
We presented a practical, dynamically activated reliability-aware algorithm that specifically considers core wear states when making assignment and scheduling decisions. Our algorithm was shown to improve system MTTF by at least 97% compared to a temperature-aware algorithm. In the future, we plan on extending our work to use DVFS for even finer-grained controls and to consider real-time systems.

References

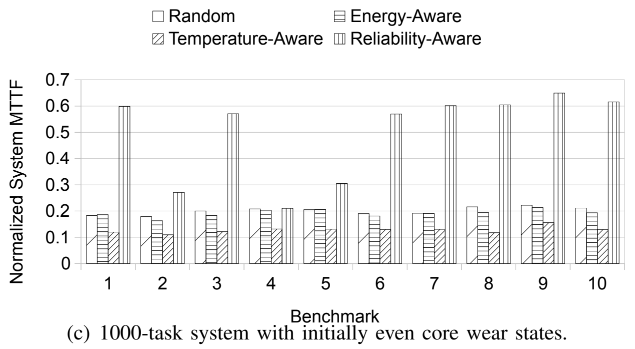
[1] R. Viswanath, et al., “Thermal performance challenges from silicon to systems,” *Intel Technology J.*, vol. 4, no. 3, pp. 1–16, Aug. 2000.



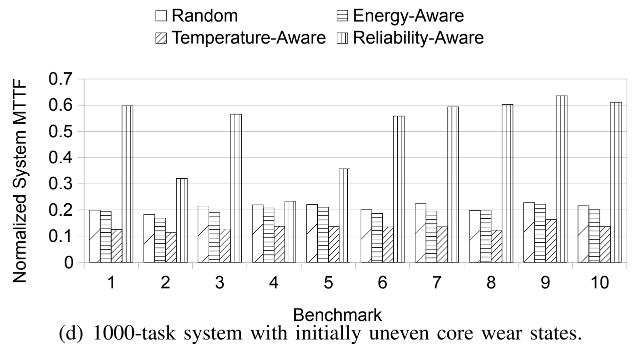
(a) 100-task system with initially even core wear states.



(b) 100-task system with initially uneven core wear states.



(c) 1000-task system with initially even core wear states.



(d) 1000-task system with initially uneven core wear states.

Fig. 3: Results for four-core systems.

[2] N. Fisher, et al., "Thermal-aware global real-time scheduling on multicore systems," in *Proc. of the Real-Time and Embedded Technology and Applications Symp.*, Apr. 2009, pp. 131–140.

[3] N. Bansal, T. Kimbrel, and K. Pruhs, "Dynamic speed scaling to manage energy and temperature," in *Proc. Symp. on Foundations of Computer Science*, Oct. 2004, pp. 520–529.

[4] J.-J. Chen, S. Wang, and L. Thiele, "Proactive speed scheduling for real-time tasks under thermal constraints," in *Proc. Real-Time and Embedded Technology and Applications Symp.*, Apr. 2009, pp. 141–150.

[5] Y. Xie and W.-L. Hung, "Temperature-aware task allocation and scheduling for embedded multiprocessor systems-on-chip (MPSoC) design," *J. VLSI Signal Processing*, vol. 45, no. 3, pp. 177–189, Dec. 2006.

[6] R. Rao and S. Vrudhula, "Performance optimal processor throttling under thermal constraints," in *Proc. Int. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems*, Oct. 2007, pp. 257–266.

[7] A. Mutapcic, et al., "Processor speed control with thermal constraints," *IEEE Trans. Circuits and Systems I*, vol. 56, no. 9, pp. 1994–2008, Sept. 2009.

[8] S. Wang and R. Bettati, "Delay analysis in temperature-constrained hard real-time systems with general task arrivals," in *Proc. Real-Time Systems Symp.*, Dec. 2006, pp. 323–332.

[9] T. Chantem, X. S. Hu, and R. P. Dick, "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.

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

[11] Z. P. Gu, et al., "Application-specific MPSoC reliability optimization," *IEEE Trans. VLSI Systems*, vol. 16, no. 5, pp. 603–608, May 2008.

[12] A. K. Coskun, et al., "Evaluating the impact of job scheduling and power management on processor lifetime for chip multiprocessors," in *Int. Conf. on Measurement and Modeling of Computer Systems*, June 2009, pp. 169–180.

[13] F. Paterna, et al., "Adaptive idleness distribution for non-uniform aging tolerance in multiprocessor system-on-chip," in *Proc. Design, Automation & Test in Europe Conf.*, Apr. 2009, pp. 906–909.

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

[15] J. Blome, et al., "Self-calibrating online wearout detection," in *Proc. Int. Symp. Microarchitecture*, Dec. 2007, pp. 109–122.

[16] M. Agarwal, et al., "Self-calibrating online wearout detection," in *Proc. Int. Test Conf.*, Oct. 2008, pp. 1–10.

[17] "Failure mechanisms and models for semiconductor devices," Joint Electron Device Engineering Council, Tech. Rep., Aug. 2003, JEP 122-B.

[18] J. R. Black, "Electromigration–a brief survey and some recent results," *IEEE Trans. Electron Devices*, vol. 16, no. 4, pp. 338–347, Apr. 1969.

[19] J. Srinivasan, et al., "Exploiting structural duplication for lifetime reliability enhancement," in *Proc. Int. Symp. Computer Architecture*, June 2005, pp. 520–531.

[20] M. Ciappa, F. Carbognani, and W. Fichtner, "Lifetime prediction and design of reliability tests for high-power devices in automotive applications," *IEEE Trans. Device and Materials Reliability*, vol. 3, no. 4, pp. 191–196, Dec. 2003.

[21] T. Chantem, et al., "Enhancing multicore reliability through wear compensation in online assignment and scheduling," Utah State University, Tech. Rep., Dec. 2013.

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

[23] J.-J. Chen, et al., "Approximation algorithms for multiprocessor energy-efficient scheduling of periodic real-time tasks with uncertain task execution time," in *Proc. of the Real-Time and Embedded Technology and Applications Symp.*, Apr. 2008, pp. 13–23.

[24] S. D. Downing and D. F. Socie, "Simple rainflow counting algorithms," *Int. J. of Fatigue*, vol. 4, no. 1, pp. 31–40, Jan. 1983.

[25] H. Huang, et al., "Throughput maximization for periodic real-time systems under the maximal temperature constraint," in *Proc. Design Automation Conf.*, June 2011, pp. 363–368.

[26] HotSpot 5.0. Computer Science Department, University of Virginia. http://lava.cs.virginia.edu/HotSpot.