Temperature Management in Multiprocessor SoCs Using Online Learning

Ayse Kivilcim Coskun[†] Tajana Simunic Rosing[†] Kenny C. Gross[‡]
[†]University of California, San Diego [‡]Sun Microsystems, San Diego

Abstract— **In deep submicron circuits, thermal hot spots and high temperature gradients increase the cooling costs, and degrade reliability and performance. In this paper, we propose a low-cost temperature management strategy for multicore systems to reduce the adverse effects of hot spots and temperature variations. Our technique utilizes online learning to select the best policy for the current workload characteristics among a given set of *expert* policies. We achieve 20% and 60% average decrease in the frequency of hot spots and thermal cycles respectively in comparison to the best performing expert, and reduce the spatial gradients to below 5%.**

Categories and Subject Descriptors: B.8 [**Performance and Reliability**]: General; C.4 [**Computer Systems Organization**]: Performance of Systems.
General Terms: Management, Design, Reliability.
Keywords: Thermal Management, Multiprocessor, Online Learning.

I. Introduction

As power consumption of chips has been increasing significantly due to technology scaling, thermal hot spots and high temperature gradients have become major challenges in multiprocessor system-on-a-chip (MPSoC) design, as they degrade reliability and performance, increase cooling costs and complicate circuit design. The number of cores on a single die increases with new generations of computers, requiring strategies to address the temperature induced challenges in a cost-effective way.

Thermal hot spots increase the cooling costs, and in addition, accelerate failure mechanisms such as electromigration, stress migration, and dielectric breakdown, which cause permanent device failures [9]. Leakage is exponentially related to temperature, so high temperatures increase leakage power. Temperature also affects performance, as the effective operating speed of devices decreases with higher temperatures. For these reasons, dynamic thermal management techniques that have been proposed in the literature generally focus on keeping the temperature below a critical threshold (e.g. [15], [4]). Such techniques prevent thermal hot spots typically at a considerable performance cost. Moreover, as dynamic thermal management techniques do not focus on balancing the temperature across the chip, they can create large spatial variations in temperature. Spatial temperature variations lead to performance degradation or logic failures, and decrease the efficiency of cooling. In process technologies below 0.13 μm, reliability issues arise due to negative bias temperature instability (NBTI) and hot carrier injection (HCI) [10]. Global clock networks are especially vulnerable to spatial variations due to clock skew [1]. Another issue with the dynamic thermal or power management methods is that, they do not prevent thermal cycling. High magnitude and frequency of thermal cycles (i.e. temporal fluctuations) cause package fatigue and plastic deformations, and lead to permanent failures [9]. Thermal cycling gets accelerated by dynamic power management (DPM) methods that turn off cores in addition to low-frequency power changes (i.e. system power on/off) [13].

Some of the reliability challenges discussed above have been addressed to some extent by techniques that perform reliability aware voltage/frequency selection at design stage [17], or that optimize the power management policy for a given reliability constraint [13]. However, as workload is hard to predict *a priori* for most systems, adaptive management strategies are required. Dynamic temperature aware scheduling techniques (e.g. [2]) proposed previously are able to reduce temporal and spatial variations as well as hot spots in comparison to conventional thermal or power management for the average case. However they do not guarantee to be effective for all execution periods, considering the trade-off between temperature and performance varies for different workloads.

The policies proposed in the literature have different optimization goals; thus, their advantages vary in terms of saving power, achieving better temperature profiles or increasing performance. For example, DPM can reduce the thermal hot spots while saving power. However, when there are frequent workload arrivals, it can significantly increase thermal cycling. Migrating threads upon reaching a critical temperature achieves significant reduction in hot spots. On the other hand, this strategy does not balance the workload across the chip.

In this work, we propose using online learning to adapt to dynamically changing workload, and to select a policy that provides the desired trade-off between performance and thermal profile. Our technique keeps a set of *expert* policies, representative of the recently proposed policies in the literature, i.e. dynamic power management (DPM), dynamic voltage-frequency scaling (DVS), thread migration, load balancing and *Adaptive-Random* [2]. Other expert policies can be integrated as needed. During execution, we monitor the runtime behavior of the currently active expert. After each time interval, we compute a multivariate loss function, which provides feedback on the temperature profile and the performance cost. The evaluation of the temperature profile takes into account the hot spots, thermal cycles and spatial gradients. We use the *switching experts* framework proposed in [5] for deciding which expert to run at each interval. Our technique is guaranteed to converge to the policy satisfying the desired trade-offs for the current workload. In our experiments we used an UltraSPARC T1 [12]. The thermal behavior and performance of our technique are evaluated using real life workloads as measured by the Continuous System Telemetry Harness (CSTH) [7]. We achieve 20% and 60% decrease on average in the frequency of hot spots and thermal cycles, respectively, in comparison to the best performing policy, and reduce the spatial gradients to below 5%.

II. Related Work

In this section, we briefly discuss the techniques for multicore scheduling and thermal management. Several power and performance aware MPSoC scheduling techniques have been proposed in the literature (e.g. [14] and [19]). As power-aware policies are not always sufficient to prevent temperature induced problems, static and dynamic thermal management methods have been proposed. Static methods for thermal and reliability management are based on thermal characterization at design time. RAMP provides an architecture level reliability model for temperature related intrinsic hard failures [17], and optimizes the architectural configuration and voltage/frequency setting for reliable design. In [13], it is shown that aggressive power management can adversely affect reliability due to fast thermal cycles, and the authors propose an optimization method for MPSoCs that saves power while meeting reliability constraints.

Dynamic thermal management controls over-heating by keeping the temperature below a critical threshold. Computation migration and fetch toggling are examples of such techniques [15]. Heat-and-Run performs temperature-aware thread assignment and migration for multicore multithreaded systems [6]. Kumar et al. propose a hybrid method that coordinates clock gating and software thermal management techniques such as temperature-aware priority management [11]. The multicore thermal management method introduced in [4] combines distributed DVS with process migration. The temperature-aware task scheduling method proposed in [2] achieves better thermal profiles than convention thermal management techniques without introducing a noticeable impact on performance.

The previously proposed power or thermal management techniques have different optimization goals and therefore work efficiently for different types of workload. In this work we introduce an online learning method that converges to the policy that provides the desired temperature/performance trade-off for the current workload. During online evaluation, we take both the thermal profile and performance into account simultaneously. This way, we demonstrate lower and more stable temperature profiles and further improvements to overall energy utilization at low performance cost. Previously, Dhiman et al. has utilized online learning for performing dynamic voltage scaling [3]. As opposed to their online learning algorithm, which evaluates each expert at every scheduler tick, we instead evaluate only a subset of policies responsible for selecting which expert to run. In addition, we focus on optimizing for temperature and performance, while the focus of [3] is reducing power consumption.

III. Online Learning for Temperature Management

A number of strategies exist to manage temperature, reduce power consumption or to perform task allocation in a temperature or power-aware manner. In most systems, workload varies dynamically at runtime, which requires using adaptive policies to find the desired trade-off between temperature and performance. In this work we propose a novel technique to adapt the thermal management policy to the current workload characteristics. The online learning framework we use, which is based on the *switching experts problem* introduced in [5], converges to the best policy for the current system dynamics from a given set of policies. In the switching experts problem, there are N *expert* policies. Also, a set of M higher level experts, called *specialists* are defined. A specialist is a higher level policy that determines which expert should run for the next interval. For example, one specialist can choose to run DPM for all intervals, while another one can select a different expert depending on whether the system has high, medium or low utilization.

The pseudo-code for our algorithm is provided in Table I. In our technique, at any given time, only one of the experts is active. The decision to switch to another expert (or continue with the current one) is performed at every interval by the specialist currently responsible for decision making. We maintain weight vectors for the specialists, which get updated at every interval based on the observed *loss*. Loss is a non-negative value demonstrating how well a policy performs in terms of the given objective. At every decision point, the specialist with the highest weight factor is selected by the learning algorithm. This way, our technique guarantees converging to the best available policy for the current workload.

We define N experts and M specialists. Our algorithm maintains a weight vector for all the specialists, $w = < w_1, w_2, ...w_M >$. Each weight w_i represents the suitability of the specialist to the current workload characteristics. We initialize the w_i values by assigning equal weights, $w_i = 1/M$. As shown in the pseudocode, the weights

TABLE I. Pseudo-Code for the Online Learning Algorithm

```
Initialize $w_i = 1/M$ for $i = \{1, 2, ...M\}$
Do for $t = 1, 2, ..., U$
  1. Pick the specialist with the highest $w_i$, and
     run the expert policy determined by that specialist
  2. Compute the loss function for the last interval ($L_t$)
  3. Update the weights for the specialists associated
     with the active expert:
       $w_i^{new} = w_i^{old} e^{-n \cdot L_t}$    if active
       $w_i^{new} = w_i^{old}$                otherwise
```

are updated based on L_t, the loss observed during the last interval. At each iteration, we only update the weights of the specialists that are associated with the active ground expert. For example, if the active expert policy has been *DPM* for the last interval, we only update the weights of the specialists that would have selected DPM for that interval. Equation 1 shows the update function. We use an exponential function to update the weights as in [8]. L_t is the total loss computed during the last interval, i.e. $[t-1, t)$, and n is the learning rate. How to select n is explained in detail in [8]. In our experiments, we set $n = 0.75$. As the new weight, w_i^{new}, depends on the previous weight, w_i^{old}, weight update equation contains the history of updates.

$$w_i^{new} = w_i^{old} e^{-n \cdot L_t} \tag{1}$$

Our loss function takes both temperature and performance characteristics into account. For evaluating the reliability impact of hot spots, observing only the peak or average temperature does not provide a good intuition of the thermal behavior. For this reason, we use the "time spent above temperature threshold" metric (t_{HS}) to capture the impact of hot spots. For thermal cycles, on each core we compute the percentage of time that cycles larger than a given ΔT value are observed (t_{TC}). Similarly, for spatial gradients, we calculate the time during which large gradients occur (t_{SP}).

The components of the loss function are provided in Table II. To compute the total loss, we normalize each term and then sum all the terms. We use the "Load Average" metric (i.e. LA in Table II) to evaluate the performance cost. We did not use metrics such as IPC or CPI since they are application dependent, and it is not possible to set a threshold value to compute the performance loss. Load average is the sum of run queue length and number of jobs currently running. Therefore, if this number is low (i.e. typically below 3 or 5, depending on the system), the response time of the system is fast. If this number is getting higher, it means that the performance is getting worse. LA_c and LA_t are the load average for the last interval and the threshold load average, respectively.

TABLE II. Loss Function

Category	Amount of Loss
Hot Spots	t_{HS} (if $t_{HS} > 0$); 0 (ow*) **otherwise*
Thermal Cycles	t_{TC} (if $t_{TC} > 0$); 0 (ow)
Spatial Gradients	t_{SP} (if $t_{SP} > 0$); 0 (ow)
Performance	$(LA_c - LA_t)$ (if $LA_c > LA_t$); 0 (ow)

Convergence Bound: Another method for selecting experts is using insomniac algorithms, where there is a master algorithm that evaluates a given set of experts by comparing the performance of each expert to that of the *best* expert. The online learning technique proposed in [3] for DVS is an example of insomniac learning. However for thermal management, evaluating each expert at every interval is infeasible. Temperature of a unit is dependent on the instant power consumption of that unit, as well as the recent temperature history and the power consumption of neighboring units. Therefore, evaluating each expert's thermal behavior and performance accurately would require running each expert on a separate system in parallel. This is obviously an extremely inefficient approach. Thus, we have utilized the *specialist* approach proposed for the switching experts problem in [5]. Our algorithm evaluates a subset of *active* specialists at each iteration.

The conventional solutions to the switching experts problem require evaluating each of the exponentially many specialists at every iteration (such as in [8]), which is an insomniac learning approach. Provided that U (number of iterations in the sequence) and k (number of intervals) are known, if we could keep a weight for all possible exponentially many specialists, the total loss with respect to the best specialist is upper bounded by $k\ lnN + (k-1)ln(U/k)$ [5]. This bound is a very tight bound for this problem; however, this algorithm is computationally very costly. The switching experts framework proposed in [5] overcomes this problem by constructing *specialists* for each expert and interval, and by evaluating only the *active* specialists at each iteration. The bound achieved by the specialist algorithm is $k(lnU + o(lnU))$ larger than the bound above. As $N << U$, this bound gives a convergence rate of $O(k\ lnU/U)$.

IV. Experimental Results

Our experimental results are based on the UltraSPARC T1 processor (floorplan shown in [12]). The average power consumption (including leakage) and area distribution of the units on the chip are provided in [2]. In our experiments, we leveraged the Continuous System Telemetry Harness (CSTH) [7] to gather detailed workload characteristics of real applications. We sampled the utilization percentage for each hardware thread at every second using `mpstat`. Half an hour long traces were collected for each benchmark. To determine the active/idle time slots of cores more accurately, we recorded the length of user and kernel threads using `DTrace`. We also recorded the cache misses and floating point (FP) instructions (per 100K instructions) using `cpustat`.

We ran web server, database, compiler (gcc), compression (gzip) and multimedia benchmarks. To generate web server workload, we ran SLAMD [16] with 20 and 40 threads to achieve medium and high utilization, respectively. For database applications, we tested MySQL using `sysbench` for a table with 1 million rows and 100 threads. For multimedia benchmarks, we ran MPlayer (integer) with 640x272 video files. We summarize the details of our benchmarks in Table III.

TABLE III. Workload characteristics

Benchmark		Avg.Util(%)	L2 I-Miss	L2 D-Miss	FP instr
1	Web-med	53.12	12.9	167.7	31.2
2	Web-high	92.87	67.6	288.7	31.2
3	Database	17.75	6.5	102.3	5.9
4	Web & DB	75.12	21.5	115.3	24.1
5	gcc	15.25	31.7	96.2	18.1
6	gzip	9	2	57	0.2
7	MPlayer	6.5	9.6	136	1
8	MPl.&Web	26.62	9.1	66.8	29.9

Peak power consumption of SPARC is similar to the average power [12], so we assumed that the instantaneous power consumption is equal to the average power at each state (active, idle, sleep). We estimated the power at lower voltage levels based on the equation $P \propto f.V^2$. We assumed three built-in voltage/frequency settings in our simulations. To account for the leakage power, we used the polynomial model proposed in [18], and determined the coefficients in the model empirically to match the normalized leakage values in [18]. We used a sleep state power of 0.02 Watts, which is estimated based on sleep power of similar cores. For the crossbar, we used a simple power model, where the power consumption scales according to the number of active cores and memory accesses.

We used HotSpot 4.0 [15] for thermal modeling, and modified the simulator with UltraSPARC T1 characteristics. Thermal simulations were sampled at every 100 ms, which provided a good precision. We initialized HotSpot with steady state temperature.

We select a set of expert policies representative of the recently proposed power and thermal management approaches, covering a variety of trade-off points among temperature, performance and power. The **Default** policy is a performance-oriented policy similar to schedulers in modern OSes. It tries to allocate threads to the same core they ran previously in order to optimize memory accesses. Load balancing is performed if there is congestion. **Dynamic power management (DPM)** turns off idle cores based on a fixed timeout strategy. **DVS & DPM** applies a dynamic voltage/frequency scaling policy which reduces the V/f level depending on the utilization observed on each core in the last interval. Idle cores are turned off using a fixed timeout policy as in DPM. **Migration** moves threads from hot cores to cooler cores when a temperature threshold is exceeded. DPM, DVS & DPM and Migration run together with the Default scheduling policy. We also use the **Adaptive-Random** policy proposed in [2] as one of the experts. Instead of keeping a set of specialists covering all possible segmentations of experts, we use the following specialists. In addition to the specialists that run the same expert all the time (i.e. Default, DPM, DPM&DVS, Migration, Adaptive-Random), to provide faster convergence to the best available policy, we developed specialists that select an expert based on system characteristics. These specialists are:

- Utilization based: We observe the average system utilization and select the expert based on the following rules:
  ** High Util.: Migration*
  ** Medium Util.: Adaptive-Random and DVS&DPM*
  ** Low Util.: DPM*
- Temperature based: Based on the thermal profile observed in the last interval, we select the policy for the next interval.
  ** Hot spots or variations: Adaptive-Random*
  ** Otherwise: Default policy*

The loss function has four components that address hot spots, cycles, spatial gradients and performance. Figure 1-(a) shows how weighing these components in the loss function changes the frequency each expert is selected. "All equal" assigns equal weights to all components, "P-high" assigns a higher weight to the performance loss, "P-low" assigns higher weight to temperature-related loss and "w/o gradients" compute loss based only on hot spots and performance, without considering gradients. It can be seen that "P-high" selects the policies with minimal performance cost more frequently than others, whereas the "all equal" setting favors Adaptive-Random and Migration more often. DPM and DVS&DPM are typically selected less often than other experts, as they create cycles. In the "w/o gradients" setting, we see a significant increase in the frequencies of selecting DPM and DVS&DPM. In the rest of our evaluation, we use the "all equal" loss function, as we want to minimize all the temperature-induced problems at low performance cost.

In our experimental evaluation, the hot spot results demonstrate the percentage of time spent above 85^oC, which is considered a high temperature for our system. The spatial gradient results summarize the percentage of time that gradients above 15^oC occur, as gradients of $15-20^oC$ start causing clock skew and delay issues [1]. The spatial distribution is calculated by evaluating the temperature difference between hottest and coolest cores at each sampling interval. For metallic structures, assuming the same frequency of thermal cycles, when ΔT increases from 10 to 20^oC, failures happen 16 times more frequently [9]. So, we report the temporal fluctuations of magnitude above 20^oC. ΔT values we report are computed over a sliding window and averaged over all cores.

In Figure 1-(b), we demonstrate how each expert behaves in terms of handling the thermal hot spots and temperature variations, and we compare their performance. These results are averaged over the benchmarks in Table III. We only show thermal cycling results for

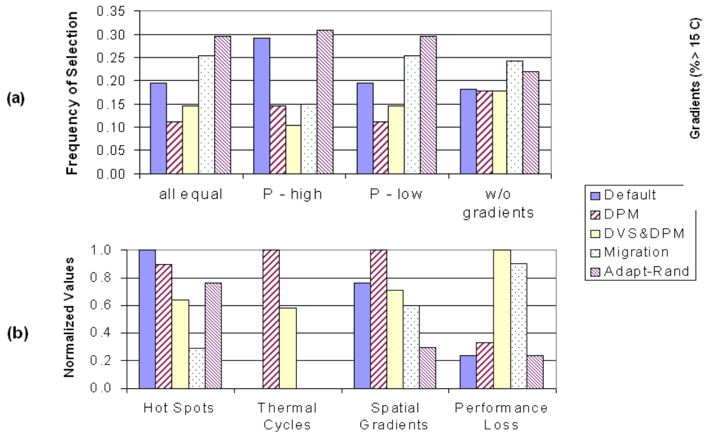

Fig. 1. (a) Effect of Loss Function on Expert Selection, (b) Evaluation of Expert Strategies

the experts with DPM, as going into the sleep state causes cycles with high magnitudes. We normalized all values to [0,1] for the sake of comparison. In the figure, we observe that our experts have different strengths. For example, `Migration` reduces the thermal hot spots more efficiently than other techniques, however it causes higher performance cost. `DPM` decreases the hot spots and does not have high performance cost, but it causes cycles and gradients.

To show how the online learning method adapts to varying workload behavior and compare it against other strategies, we ran sequences of the benchmarks in Table III. In our results, the sequences are identified by the numbers associated with each benchmark as shown in Table IV. For example, the workload sequence A runs Web-med (1) followed by Web-high(2) and so on. In Table IV, we compare the efficiency of our online learning (i.e. *OL*) technique against running each expert alone. For workloads A, B and C, OL reduces the frequency hot spots more than all of the individual experts, as it can combine the advantages of different experts over different execution intervals. For workload D, OL performs almost as good as DPM&DVS. OL reduces the frequency of hot spots by about 20% on average in comparison to DVS&DPM.

TABLE IV. Thermal Hot Spots

W.load	Util (%)	Def.	Migr.	DPM	DVS& DPM	Adapt -Rand	OL
A: 12784	50.8	18.6	11.4	16.9	8.9	13.2	6.5
B: 57843	28.2	8.4	4.2	6.4	2.3	5.4	2.1
C: 14214	69.8	27.8	18.1	21.3	14.9	19.2	9.7
D: 68253	32.3	10.3	5.7	8.0	2.7	6.4	2.9

Figures 2 (a) and (b) show how the online learning method compares to other methods in terms of reducing temperature variations. Our method reduces the cycles dramatically, i.e. 80% and 68% in comparison to DPM and DVS&DPM, respectively. OL is close to Adaptive-Random in terms of reducing the spatial gradients. We cannot achieve significant reduction of gradients in comparison to Adaptive-Random, because to reduce the hot spots more effectively, our policy sometimes favors other policies.

DPM and DVS&DPM reduce the power consumption by about 13% and 16%, respectively, in comparison to the Default policy. The online learning technique achieves close to 6% of power savings. As Migration and Adaptive-Random are not integrated with DPM or DVS in our work, they do not reduce power consumption. We also computed the performance cost based on the average wait time for jobs on the system. The average wait time of jobs are increased by 2.1 times for Migration, 1.3 times for DPM, and 3.2 times for DPM&DVS. Adaptive-Random does not increase the wait time,

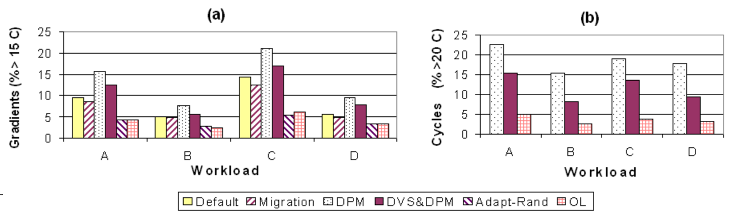

Fig. 2. (a) Thermal Cycles, (b) Spatial Gradients

and OL only increases it by 1.1 times. Thus, while our technique decreases the temperature induced problems at low performance cost, we can also save about 6% power on average.

V. Conclusion

We have presented a MPSoC temperature management technique that utilizes online learning to adapt to dynamically changing workloads. Online learning evaluates the management policies at runtime by taking both thermal behavior and performance into account, and guarantees convergence to the most beneficial policy for the desired performance-temperature trade-off. We have shown that our technique reduces the hot spots, thermal cycles and gradients significantly more than running a single temperature or power aware policy.

Acknowledgement

This work has been funded by Sun Microsystems, and the University of California MICRO grant 06-198.

References

[1] A. Ajami, K. Banerjee, and M. Pedram. Modeling and analysis of nonuniform substrate temperature effects on global ULSI interconnects. *IEEE Transactions on CAD*, 24(6):849–861, June 2005.
[2] A. K. Coskun, T. Rosing, and K. Whisnant. Temperature aware task scheduling in MPSoCs. In *DATE*, 2007.
[3] G. Dhiman and T. Rosing. Dynamic voltage frequency scaling for multitasking systems using online learning. In *ICCAD*, 2007.
[4] J. Donald and M. Martonosi. Techniques for multicore thermal management: Classification and new exploration. In *ISCA*, 2006.
[5] Y. Freund, R. E. Schapire, Y. Singer, and M. K. Warmuth. Using and combining predictors that specialize. In *STOC*, 1997.
[6] M. Gomaa, M. D. Powell, and T. N. Vijaykumar. Heat-and-Run: leveraging SMT and CMP to manage power density through the operating system. In *ASPLOS*, 2004.
[7] K. Gross, K. Whisnant, and A. Urmanov. Electronic prognostics through continuous system telemetry. In *MFPT*, pages 53–62, April 2006.
[8] M. Herbster and M. K. Warmuth. Tracking the best expert. In *International Conference on Machine Learning*, pages 286–294, 1995.
[9] Failure mechanisms and models for semiconductor devices, JEDEC publication JEP122C. http://www.jedec.org.
[10] H. Kufluoglu and M. A. Alam. A computational model of NBTI and hot carrier injection time-exponents for MOSFET reliability. *Journal of Computational Electronics*, 3 (3):165–169, Oct. 2004.
[11] A. Kumar, L. Shang, L.-S. Peh, and N. K. Jha. HybDTM: a coordinated hardware-software approach for dynamic thermal management. In *DAC*, pages 548–553, 2006.
[12] A. Leon, L. Jinuk, K. Tam, W. Bryg, F. Schumacher, P. Kongetira, D. Weisner, and A. Strong. A power-efficient high-throughput 32-thread SPARC processor. *ISSCC*, 2006.
[13] T. S. Rosing, K. Mihic, and G. D. Micheli. Power and reliability management of SoCs. *IEEE Transactions on VLSI*, 15(4), April 2007.
[14] M. Ruggiero, A. Guerri, D. Bertozzi, F. Poletti, and M. Milano. Communication-aware allocation and scheduling framework for stream-oriented multi-processor system-on-chip. In *DATE*, 2006.
[15] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *ISCA*, 2003.
[16] SLAMD Distributed Load Engine. www.slamd.com.
[17] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. The case for lifetime reliability-aware microprocessors. In *ISCA*, 2004.
[18] H. Su, F. Liu, A. Devgan, E. Acar, and S. Nassif. Full-chip leakage estimation considering power supply and temperature variations. In *ISLPED*, 2003.
[19] Y. Zhang, X. S. Hu, and D. Z. Chen. Task scheduling and voltage selection for energy minimization. In *DAC*, 2002.