Temperature Aware Task Scheduling in MPSoCs

Ayse Kivilcim Coskun[†] Tajana Simunic Rosing[†] Keith Whisnant[‡]
[†]University of California, San Diego [‡]Sun Microsystems, San Diego

Abstract

In deep submicron circuits, elevation in temperatures has brought new challenges in reliability, timing, performance, cooling costs and leakage power. Conventional thermal management techniques sacrifice performance to control the thermal behavior by slowing down or turning off the processors when a critical temperature threshold is exceeded. Moreover, studies have shown that in addition to high temperatures, temporal and spatial variations in temperature impact system reliability. In this work, we explore the benefits of thermally aware task scheduling for multiprocessor systems-on-a-chip (MPSoC). We design and evaluate OS-level dynamic scheduling policies with negligible performance overhead. We show that, using simple to implement policies that make decisions based on temperature measurements, better temporal and spatial thermal profiles can be achieved in comparison to state-of-art schedulers. We also enhance reactive strategies such as dynamic thread migration with our scheduling policies. This way, hot spots and temperature variations are decreased, and the performance cost is significantly reduced.

1 Introduction

Advancements in process technology enable manufacturing a complete MPSoC including CPUs, memories and communication architectures on a single die. Sun's UltraSPARC T1 [14] and IBM's Cell [13] are examples of such systems. However, as semiconductor processing technology migrates to more advanced deep-submicron technologies, new challenges arise for the next generation of MPSoC products. High power and temperature densities, process imperfections and reduced voltage margins have made the systems much more vulnerable to both permanent and transient faults. Elevation in temperatures (i.e. thermal hot spots) and temperature gradients bring new challenges in reliability, timing, performance, cooling costs and leakage power. In this work, we propose MPSoC scheduling optimization with negligible performance overhead for mitigating these temperature induced problems.

Thermal hot spots cause high cooling costs, poor reliability and performance degradation. Cost of cooling increase at a super-linear rate [9], which requires designing for temperature margins that are lower than the worst-case. Hot spots accelerate the failure mechanisms such as electromigration, stress migration and dielectric breakdown, which cause permanent device failures [12]. In fact, a small difference in the operating temperature (i.e. $10 - 15^oC$) can result in a 2X difference in the lifespan of the devices [29]. Leakage is exponentially related to temperature, and a positive feedback loop exists between temperature and leakage, which can cause dramatic increases in temperature and damage the circuit if not controlled. For feature sizes below 65nm, leakage is expected to account for more than 50% of the overall power consumption [23]. High temperatures can also adversely affect performance, because the effective operating speed decreases with high temperatures due to the carrier mobility's dependence on temperature.

In addition to thermal hot spots, temporal fluctuations in temperature have an adverse effect on system reliability. Thermal cycling phenomenon is correlated with the magnitude and frequency of the temperature cycles the device experiences [12]. It causes accelerated package fatigue and plastic deformations of materials that accumulate at each temperature cycle, and can lead to cracks and other permanent failures. Temperature cycles are created by either low frequency power changes such as system power on/off cycles, or workload rate changes and power management decisions which happen much more frequently [24]. Previous work shows that addressing thermal hot spots alone is not enough to achieve better reliability, and temperature gradients in time and space determine device reliability at moderate temperatures [16].

Spatial temperature variations across the chip can cause performance mismatches, leading to performance or logic failures. Negative bias temperature instability (NBTI) and hot carrier injection (HCI) cause the circuits to fail in meeting the timing constraints [15]. In process technologies below 0.13um, reliability issues arise due to NBTI and HCI, considering that the operating temperatures and electric fields reach high enough values to accelerate these mechanisms during device lifetime. A number of other design issues arise due to large spatial differentials. Local resistances scale linearly with temperature. Increasing temperature increases resistances, and thus circuit delays and IR drop [23]. Global clock networks are especially vulnerable to spatial variations. Every 20 degrees increase in temperature causes 5-6% increase in Elmore delay in interconnects. As a result, clock skew problems become noticeable for spatial variations of around 20 degrees and above [2]. Recently, there have been proposals that address such delay problems in clock networks (e.g. [6, 2]).

To date, the temperature related problems have been addressed to some extent by techniques that lower the average temperature or keep the temperature under a given threshold. Power aware synthesis, dynamic power management (DPM), and dynamic voltage scaling (DVS) are such techniques. Thermal modeling and management have been introduced to guarantee the die temperature does not reach critical values. A significant bottleneck of traditional thermal management methods

978-3-9810801-2-4/DATE07 © 2007 EDAA

is the performance impact associated with stalling or slowing down the processor [26]. When the workload that is going to run on the system is known (e.g. in some embedded systems), predictive techniques can be applied which select voltage/frequency levels or architecture configuration at design stage to avoid dynamic thermal management as much as possible [27].

Despite their significant benefits to the thermal profile of the chip, conventional power or thermal management techniques cannot always eliminate the problems associated with temperature in a cost-effective way. Both temporal and spatial temperature variations cause a number of reliability problems, while typical power and thermal management do not focus on the effects of these variations. Moreover, reactive thermal management methods, which address some of the temperature problems effectively, often have considerable performance impact. In high end computing domain, lower overhead strategies such as optimizing workload and core frequency at design stage are not very useful because workload varies significantly over time. Thus, techniques that can lower and balance temperature at runtime with low performance overhead are needed.

In this paper, we investigate how dynamic OS-level workload scheduling can achieve temperature profiles that are beneficial for reliable MPSoC design. In contrast to thermal management techniques, which perform computation migration or clock gating (e.g. [25]) when temperatures reach critical values, our goal is to adjust the workload distribution to achieve the best temporal and spatial temperature distribution possible. We evaluate heuristic methods that make decisions based on the current temperature, and we also propose a probabilistic scheduling technique, *Adaptive Random*. This technique adapts to the changes in the temperature by taking into account the thermal history when making scheduling decisions. We also look at how dynamic power management affects the temperature, and provide results on combined power management/thermal management policies. Our analysis shows that, when combined with reactive methods such as thread migration and voltage scaling, the Adaptive Random policy decreases the performance impact of such techniques considerably, while achieving better thermal profiles.

2 Related Work

In this section, we first briefly discuss the previous work on multiprocessor scheduling. We then review more closely the prior work on thermal management, and compare it against our approach.

2.1 Scheduling in MPSoCs

In MPSoC research, a lot of focus has been on optimizing the scheduling with performance and energy objectives. A power management strategy for mission critical systems containing heterogeneous devices is proposed in [18]. Design time and runtime optimizations are combined in a two-level strategy introduced in [30]. The problem of concurrent communication and task scheduling for heterogeneous NoCs is formulated in [10], and a static schedule that considers real-time constraints is proposed. Rong et al. formulates the optimization problem of finding the optimal voltage schedule and task ordering for a system consisting of a single core and peripheral devices using integer linear programming (ILP), and proposes a three-phase solution framework [20]. In [21], the MPSoC scheduling problem is decomposed into allocation and scheduling sub-problems, which are solved using ILP and constraint programming respectively, with the objectives of minimizing the data transfer on the bus and guaranteeing deadlines for the average case.

In our work, we focus on MPSoCs for the high end computing domain, where workload is not known a priori and generally not easy to predict. For such systems, making dynamic and fast decisions for workload allocation is crucial for performance. Thus, the techniques we focus on are low overhead OS level schedulers, as opposed to high complexity strategies with aggressive performance and energy optimization goals.

2.2 Thermal Modeling and Management

Elevated temperatures impact system reliability, cause difficulties in circuit design and increase the cooling costs significantly. Consequently, accurate thermal modeling has become a requirement. HotSpot [26] is an automated thermal model, which calculates transient temperature response given the physical characteristics and power consumption of units on the die. A fast thermal emulation framework for FPGAs is introduced in [3], which reduces the thermal simulation time considerably while maintaining accuracy. As policies based on power metrics are not sufficient to prevent local thermal hot spots, several thermal management approaches have been presented to date. These methods are either dynamic (online) and make decisions while the system is running, or static (offline) where optimizations are performed at design stage.

Dynamic thermal management (DTM) controls overheating by keeping the temperature below a critical threshold. Computation migration and fetch toggling are examples of DTM techniques [26]. Heat-and-Run performs temperature aware thread assignment and migration for multicore multithreaded systems [7].

Static methods for thermal and reliability management are based on system characterization at design time. In [27], a thermal management approach is introduced that predicts the hot spots based on the execution profile of the multimedia benchmarks. A simulated-annealing based thermal floorplanning method at micro-architecture level is presented in [22], which achieves 20 degrees reduction of temperature in average. Including temperature as a constraint in the co-synthesis framework and in task allocation for platform-based system design is introduced in [11]. RAMP provides a reliability model at architecture level for temperature related intrinsic hard failures [28]. It analyzes the effects of application behavior on reliability, and optimizes the architectural configuration and power/thermal management policies for reliable design. The trade off between power consumption and reliability is studied in [24], and it is shown that aggressive power management can adversely affect reliability due to fast thermal cycles. They propose a joint policy optimization method that can achieve high amount of power savings while meeting the reliability criteria on MPSoCs.

Our work focuses on optimizing the thermal profile of the MPSoC with minimal impact on performance. Our technique can be combined with more aggressive DTM strategies to re-

duce their performance cost while further lowering and balancing the temperature.

3 Temperature Aware Task Scheduling

In this work, we present dynamic OS-level temperature aware scheduling techniques with negligible performance overhead. These techniques can mitigate the thermal hot spots and large temperature variations, which cause significant challenges in reliable system design. When combined with previously introduced reactive thermal management methods such as thread migration and voltage scaling, the proposed techniques can achieve even lower and more stable thermal profiles while reducing the performance impact of reactive techniques significantly. The thermally-aware policies we investigate are cost-effective and can be easily implemented into existing schedulers at the OS level.

Most of the techniques we discuss in this work make decisions based on the temperature measured on the MPSoC. Current chips typically contain several thermal sensors, and these sensors can be read by a continuous system telemetry infrastructure for collecting and analyzing time series sensor data [8]. The data from the sensors are written to memory and read by the OS to be processed on a regular basis. This information is then passed to the scheduler at each time interval to guide scheduling in a thermally-aware way. The thermal information collected by sensors often includes a certain amount of inaccuracy. However, online validation of the sensor outputs can be performed through advanced pattern recognition techniques such as the multivariate state estimation technique (MSET) [8], ensuring that only valid sensor readings drive the scheduler.

In this section, we provide the details of the scheduling techniques that were implemented for this work. We start by discussing multiprocessor schedulers in state-of-art operating systems. We then follow with two previously introduced reactive thermal management methods that are applicable to MPSoCs. Finally, we explain the temperature aware scheduling techniques we propose.

3.1 State-of-the-Art Load Balancing Schedulers

Many modern OS schedulers are based on multilevel queuing, which is a dynamic technique that mixes several elements such as priority, round-robin and shortest-job-first scheduling principles. While there are other scheduling methods such as gang scheduling (which sends related threads to same core) and dedicated processor assignment (which uses fixed processor assignments), multilevel queuing with some amount of load balancing is commonly used for performance reasons. In Linux 2.6, each processor in the multiprocessor system has a queue, and a task stays in a queue for cache affinity. Tasks are moved to different queues only when the load is unbalanced (i.e., when length of one queue is less than one fourth of another). Solaris makes use of load balancing for performance reasons, and migrates threads to other processors when a core becomes overloaded. The thread migration in Solaris is performed based on giving priority to locality, following the assumption that the threads on nearby cores share the same caches.

In this work, we implemented a dynamic strategy (referred to as *Load Bl.* in later sections) where the scheduler balances the workload by sending workload to the least busy processor at each interval. This dynamic load balancing strategy is in principal similar to load balancing performed by operating systems such as Solaris, which balances the workload in the processors' queues at regular intervals. We used a balancing interval of 200 seconds in our experiments.

3.2 Thermal Management Techniques for MPSoCs

Here we discuss two previously introduced techniques that can be applied to MPSoCs for controlling temperature. Both of these techniques are reactive, that is they are activated only when a critical temperature is reached.

Dynamic Thread Migration is an MPSoC thermal management method that migrates threads from hot processors to cooler ones. For minimizing the performance impact of thread migration, Heat-and-Run proposed loading the cores as much as possible and migrating workload when critical temperature values are observed [7]. In our implementation of this technique, we migrate the thread from the hot processor to the coolest processor available at that moment. The threshold temperature for migration is set at 85^oC, which is considered a critical temperature in many systems.

Voltage Scaling for Thermal Management (VSTM) performs dynamic voltage and frequency scaling when the temperature reaches the threshold [26]. This technique lowers the temperature on the hot cores by reducing power consumption. In our implementation, we assume two built-in voltage/frequency settings for each core. All jobs run at full speed (f_{max}) unless a critical temperature value (85^oC) is observed. If a core reaches the critical value, the voltage level of the particular core is reduced to the lower setting (f_{low}) until the current job terminates. In our experiments, f_{low} is two thirds of f_{max}.

3.3 Low-Overhead Temperature Aware Scheduling

The policies we investigate here have negligible overhead in comparison to the existing decision-making process in OS-level multiprocessor schedulers, and they can be implemented in the OS scheduler with minimal changes. We first look at heuristic methods that make scheduling decisions based on the current temperature and floorplan. We then present a more advanced approach that adjusts the likelihood of each processor receiving workload based on the temperature history.

Sending Workload to Coolest Core: Reactive methods for thermal management have to trade off performance to control temperature. The heuristics we investigate here avoid excessive heating of cores by making scheduling decisions based on the current temperature of the cores. The two heuristics we implemented are: 1) *Coolest*, where for each ready job, the scheduler selects the coolest processor for allocation; 2)*Coolest-FLP*, where the principle is same as (1), but in addition the scheduler gives priority to processors that have "idle" neighbors. Horizontal heat transfer plays an important role in determining the temperature, as discussed previously in [22]. Coolest-FLP accounts for the fact that a significant amount of heat transfer occurs among neighboring units on the die, and active processors with active neighbors will cause the particular region to heat up faster. Considering the floorplan and current activity of the neighbor units also decreases the spatial variations in temperature.

Random Policy with Temperature Aware Adaptation: The scheduling policies we have discussed so far have different strengths. For example, load balancing achieves better load distribution and higher performance. On the other hand, making scheduling decisions based on current temperature decreases the hot spots and temperature gradients. In order to address several objectives and yet avoid introducing significant complexity to the scheduler, we developed a probabilistic policy called *Adaptive-Random*. This policy updates probabilities of sending workload to cores at each interval based on an analysis of the temperature history on the chip.

At each job arrival, the new probability value for each core is computed using Equation 1. In the equation, P_n is the new probability, P_o is the previous probability, and W is the weight. P_n values saturate at 0 and 1. In order to evaluate the thermal stress on each core, W is computed at regular intervals using a sliding window of temperature history. We set the interval and sliding window lengths at 1 second in order to account for the rapid changes in temperature. As we compute only Equation 1 at workload arrivals, the computation cost of our technique is negligible, and we do not have to stall execution. Once the probabilities are updated, the core for allocating the current job is selected through generating a random number.

$$P_n = P_o \pm W \tag{1}$$

The probability values are decremented or incremented by W_{dec} or W_{inc}, depending on whether the temperature has risen above the threshold temperature (T_{thr}), or dropped below a second threshold T_{low} respectively. If there are processors that have exceeded T_{thr} in the past interval, their P_n values are dropped to 0 (i.e. $W_{dec} = P_o$). We increase the P_n of cores that did not spend any time above T_{thr} by W_{inc} (see Eqn. 2). In our simulations we set T_{thr} at 80^oC. We used a threshold lower than 85^oC for preventing hot spots before they occur. In order to avoid allocating workload to cores that have temperatures slightly below 80^oC, we used a second threshold, T_{low}. We do not increase P_n unless in the last interval the core temperature has dropped below T_{low}, which is 75^oC in our experiments. While calculating W_{inc}, we evaluate Av_{thr}, which is the average temperature below T_{thr} divided by T_{thr}. This way, if a core is cooler than another, its W_{inc} is greater. We selected the β value as 0.1 empirically. When the temperature of a core is between T_{thr} and T_{low}, no action is taken.

$$W_{inc} \quad = \beta / Av_{thr} \tag{2}$$

This scheduling policy can be implemented on a real system easily. As discussed previously, collecting the thermal data and computation of weights do not impose noticeable performance impact. The random number generator is implemented through a linear-feedback shift register (LFSR), which often already exists on the chip for test purposes. Another benefit of this policy is that it achieves better load balancing than making decisions solely on current temperature. The Adaptive-Random policy addresses the issues of maintaining a balanced and low temperature profile as well as distributing the thermal stress to cores as evenly as possible throughout system lifetime.

4 Results

4.1 Experimental Methodology

In this work, our experimental results are based on the data collected from an UltraSPARC T1 processor, which contains 8 cores and memory, communication and I/O units. This MPSoC has been manufactured in 90nm process technology. The processors are in-order execution cores and have multithreading capability. Every two cores share an L2-cache. The cores communicate through shared memory. The power distribution among the units and relative sizes of each unit on the chip are provided in Table 1. The power data is obtained through simulations. Figure 1 demonstrates the MPSoC floorplan [17].

Component Type	Power (%)	Area (%)
Cores	65.27	37.66
Caches	25.50	50.69
Crossbar	6.01	5.84
Other	3.22	5.81

Table 1. Power and area distributions of the units

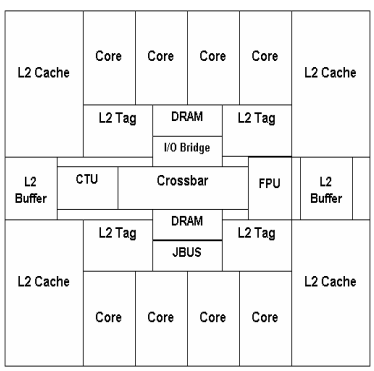

Figure 1. Floorplan of the MPSoC

We collected the percentage of times the processors were active or idle throughout the execution of a collection of CPU-intensive, network-intensive, memory-intensive and bus-intensive threads. The MPSoC runs a multilevel queuing scheduler with basic load balancing capabilities.

To fairly compare different scheduling techniques, we implemented a simulator. In our simulation, we took a representative trace of data collected at runtime. Our simulation trace is 70 minutes long, and the execution time of each job in the trace varies between 10 and 1000 seconds. In the first step of the simulator, the scheduler is provided a list of jobs and their start times, which represents the job arrivals to the system. The scheduler then dynamically allocates the ready tasks.

In the next step of the simulator, power values are derived based on each unit's execution profile. Dynamic power management or voltage scaling is also applied in this stage, depending on the policy simulated. For cores, we used average power values for the active and idle states for UltraSPARC T1. In the average case, the ratio between active and idle state power is 7.4. We estimated the power at the lower voltage levels based on the relationship between power, frequency and voltage (i.e. $P \propto f * V^2$). We used reported sleep state values for similar cores in order to estimate the sleep state power. For the

crossbar, we used a simple power model, where the power consumption scales according to how many cores are active. For dynamic power management (DPM), we implemented a fixed timeout policy [4] with timeout set to 100ms. We also apply dynamic voltage scaling as a part of the thermal management (For more details, see Section 3.2).

The next step is to obtain the temperature distributions using a thermal simulator. We used HotSpot version 2 [26] as the thermal modeling tool, and modified it accordingly for MPSoCs. For characterizing the thermal package, we used the default heat sink and spreader parameters (i.e. thermal resistance and capacitance values) in HotSpot, which represent a thermal package commonly found in modern processors. We observed the thermal constant of our system at several hundred milliseconds, and we used a time slice of 100ms in the thermal simulations which provided a good precision. We use the temperature output from HotSpot [26] to drive temperature based decisions at the next scheduling point.

Next we present the experimental evaluation of the scheduling techniques. We compare their efficiency in reducing high temperatures and temperature gradients, with and without power management. We also evaluate the performance of each technique.

4.2 Thermal Hot Spots

Here we compare the thermal distributions achieved by each technique we discussed in the previous section. In Table 2, we show the percentage of time spent above 85^oC. We provide results for cases with and without DPM. In the first row, we show the normalized performance of each policy. To evaluate the performance impact, we computed the average delay in the completion time of jobs with respect to the baseline case of load balancing. In thread migration, we assumed each migration takes 200ms, according to the results provided in [5]. All the results regarding the Adaptive-Random technique are averaged over a hundred runs in order to obtain statistical convergence.

We observe from these results that the two heuristics we proposed and the Adaptive-Random technique perform similarly with load balancing in terms of reducing the hot spots. When DPM is applied, the benefits of Coolest increase. This is due to the fact that much lower temperatures that can be achieved in the sleep state. When we combine thread migration (DTM) and voltage scaling with Adaptive-Random, the performance impact is significantly reduced, while even lower frequency of hot spots is achieved.

4.3 Spatial Temperature Variations

Load balancing balances the workload based on the processor activity in an interval, but does not consider temperature distributions. Therefore, very often large spatial differential temperatures are observed. The spatial variations across the whole chip can easily go above 20^oC. Previous work shows that clock skew resulting from spatial gradients becomes noticeable for gradients around 20 degrees and above [2]. Effects of thermal gradients on performance of devices can even be much more severe than the effect on interconnects, as it is shown that the driver on-resistance is more dependent on temperature than interconnect resistance [1]. In fact, device delay is correlated with the driver on-resistance, which increases by 30% when the

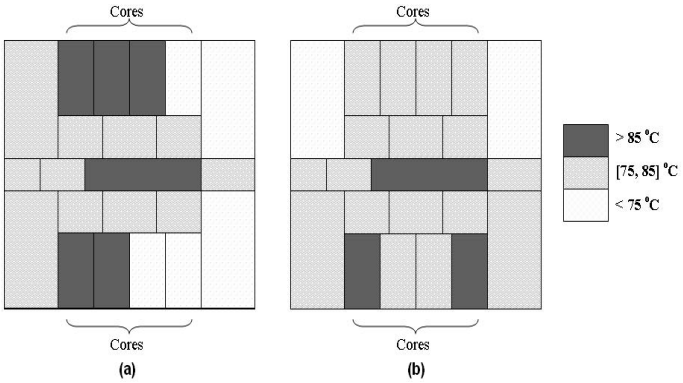

Figure 2. Thermal maps: (a)Load Bl.; (b)Adaptive-Random

temperature rises from 70^oC to 90^oC in 0.1um technology [1]. For these reasons, even at 15 degrees of gradients, performance degradation could occur due to varying delays of components.

Figure 2 shows thermal maps of MPSoCs (a) and (b), which have been scheduled with load balancing and Adaptive-Random respectively. Load balancing causes high spatial temperature differences (above 20^oC) across the chip frequently, whereas Adaptive-Random typically achieves a more uniform thermal profile. For example, without power management, we observe that load balancing causes the maximum spatial differential among the cores to stay above 10^oC 99% of the time. However, Adaptive-Random reduces this percentage to below 30%. We can observe from the figure that scheduling has a considerable effect on the spatial variations. Especially for chips with a larger number of cores, spatial gradients will introduce more challenges for system design and reliability.

4.4 Temporal Temperature Variations

Here we evaluate how each technique affects temporal temperature fluctuations on the cores. Higher magnitude and frequency of thermal cycles affect reliability adversely [12, 28]. Changes in the workload cause a certain amount of temperature variations on each core, but most of the time the magnitude of these variations are not large enough to cause problems. However, *sleep* state of the cores typically consume power much less than the active state, and dynamic power management can create large enough variations in temperature to increase the failure rates considerably [24]. Therefore, in our evaluation, we only investigate the cases with DPM.

The number of cycles to failure can be approximated as $N_f = C_o * (\Delta T)^{-q}$ using Coffin-Manson model [12], where C_o is a material dependent constant and q is an empirically derived constant. For example, if we use a q value of 4 for metallic structures [12], assuming the same frequency of cycles, when ΔT increases from 10 to 20^oC, N_f decreases by a factor of 16. Table 3 shows the average percentage of time the cores experience temporal variations above 20^oC. These results were obtained by computing the ΔT over a sliding window of 1000 seconds for each core, and averaging the ΔTs of all cores in the MPSoC.

While load balancing performs similarly to the temperature aware techniques in terms of reducing the hot spots, it causes cores to experience ΔTs greater than 20^oC close to 10% of the time. Load balancing cannot effectively take care of the temporal variations because balancing workload does not consider the

	Load Bl.	Coolest	Coolest-FLP	AdaptRand	DTM	AdaptRand-DTM	VSTM	AdaptRand-VSTM
Without DPM								
Performance	1	1	1	1	0.941	0.976	0.85	0.935
$\% > 85^oC$	1.19	1.21	1.21	1.15	0.50	0.09	0.00	0.00
With DPM								
$\% > 85^oC$	0.29	0.09	0.29	0.28	0.00	0.00	0.00	0.00

Table 2. Performance Evaluation and Hot Spots

	Load Bl.	Coolest	Coolest-FLP	AdaptRand	DTM	AdaptRand-DTM	VSTM	AdaptRand-VSTM
With DPM								
$\% > 20^oC$	9.66	5.25	5.98	4.91	2.30	0.15	5.14	0.12

Table 3. Average Temporal Temperature Variations

temporal variations on each core. With negligible performance overhead, this ratio can be reduced by almost a factor of two using thermally-aware techniques. When Adaptive-Random is combined with thread migration or voltage scaling, high temporal variations are almost completely eliminated. Studies show that having a more constant temperature will always result in a better reliability than a varying temperature with the same average value [19]. Therefore, minimizing the temporal fluctuations is a significant achievement for improving system reliability.

5 Conclusion

In this paper, we investigated dynamic OS-level schedulers for thermal management of MPSoCs. We showed that techniques that take into account temperature measurements achieve low and balanced temperature profiles at negligible performance cost. We developed an adaptive policy, which modifies the workload allocation policy based on the temperature history. While the adaptive method was slightly better in eliminating hot spots than existing load balancing techniques, it provided significant reductions in temporal and spatial temperature variations. We also demonstrated that the performance overhead of reactive techniques such as thread migration and voltage scaling can be reduced dramatically when they are combined with the Adaptive-Random technique, while achieving lower and more stable temperatures.

6 Acknowledgements

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

References

[1] A. H. Ajami, K. Banerjee, and M. Pedram. Analysis of substrate thermal gradient effects on optimal buffer insertion. In *ICCAD*, pages 44–48, 2001.

[2] A. H. 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.

[3] D. Atienza, P. D. Valle, G. Paci, F. Poletti, L. Benini, G. D. Micheli, and J. M. Mendias. A fast HW/SW FPGA-based thermal emulation framework for multi-processor system-on-chip. In *DAC*, 2006.

[4] L. Benini, A. Bogliolo, and G. D. Micheli. A survey of design techniques for system-level dynamic power management. *IEEE Trans. Very Large Scale Integr. Syst.*, 8(3):299–316, 2000.

[5] S. Bouchenak and D. Hagimont. Pickling threads state in the java system. In *Technology of Object-Oriented Languages and Systems (TOOLS) Europe*, 2000.

[6] A. Chakraborty, P. Sithambaram, K. Duraisami, A. Macii, E. Macii, and M. Poncino. Thermal resilient bounded-skew clock tree optimization methodology. In *DATE*, 2006.

[7] 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.

[8] K. Gross, K. Whisnant, and A. Urmanov. Electronic prognostics through continuous system telemetry. In *60th Meeting of the Society for Machine Failure Prevention Technology (MFPT)*, pages 53–62, April 2006.

[9] S. Gunther, F. Binns, D. Carmean, and J. Hall. Managing the impact of increasing microprocessor power consumption. *Intel Technology Journal*, 2001.

[10] J. Hu and R. Marculescu. Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints. In *DATE*, 2004.

[11] W.-L. Hung, Y. Xie, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin. Thermal-aware task allocation and scheduling for embedded systems. In *DATE*, 2005.

[12] Failure mechanisms and models for semiconductor devices, JEDEC publication JEP122C. http://www.jedec.org.

[13] J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy. Introduction to the Cell multiprocessor. *IBM Journal of Research & Development*, 49(4/5):589–604, July/September 2005.

[14] P. Kongetira, K. Aingaran, and K. Olukotun. Niagara: A 32-way multithreaded SPARC processor. *IEEE Micro*, 25(2):21–29, 2005.

[15] 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.

[16] C. J. Lasance. Thermally driven reliability issues in microelectronic systems: status-quo and challenges. *Microelectronics Reliability*, 43:1969–1974, 2003.

[17] 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.

[18] J. Liu, P. H. Chou, N. Bagherzadeh, and F. Kurdahi. Power-aware scheduling under timing constraints for mission-critical embedded systems. In *DAC*, 2001.

[19] Z. Lu, W. Huang, S. Ghosh, J. Lach, M. Stan, and K. Skadron. Analysis of temporal and spatial temperature gradients for IC reliability. *University of Virginia Technical Report CS-2004-08*, March 2004.

[20] P. Rong and M. Pedram. Power-aware scheduling and dynamic voltage setting for tasks running on a hard real-time system. In *ASPDAC*, 2006.

[21] 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.

[22] K. Sankaranarayanan, S. Velusamy, M. Stan, and K. Skadron. A case for thermal-aware floorplanning at the microarchitectural level. *The Journal of Instruction-Level Parallelism*, 7, 2005.

[23] M. Santarini. Thermal integrity: A must for low-power IC digital design. *EDN*, pages 37–42, Sept. 2005.

[24] T. Simunic, K. Mihic, and G. D. Micheli. Optimization of reliability and power consumption in systems on a chip. In *PATMOS*, 2005.

[25] K. Skadron. Hybrid architectural dynamic thermal management. In *DATE*, 2004.

[26] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *ISCA*, 2003.

[27] J. Srinivasan and S. V. Adve. Predictive dynamic thermal management for multimedia applications. In *ICS*, 2003.

[28] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. The case for lifetime reliability-aware microprocessors. In *ISCA*, 2004.

[29] R. Viswanath, V. Wakharkar, A. Watwe, and V. Lebonheur. Thermal performance challenges from silicon to systems. *Intel Technology Journal*, (Q3), 2000.

[30] P. Yang, C. Wong, P. Marchal, F. Catthoor, D. Desmet, D. Verkest, and R. Lauwereins. Energy-aware runtime scheduling for embedded-multiprocessor SoCs. *IEEE Des. Test*, 18(5):46–58, 2001.