Task Migrations for Distributed Thermal Management Considering Transient Effects

Zao Liu, Sheldon X.-D. Tan, Xin Huang, and Hai Wang

***Abstract*— In this brief, a new distributed thermal management scheme using task migrations based on a new temperature metric called effective initial temperature is proposed to reduce the on-chip temperature variance and the occurrence of hot spots for many-core microprocessors. The new temperature metric derived from frequency domain moment matching technique incorporates both initial temperature and other transient effects to make optimized task migration decisions, which leads to more effective reduction of hot spots in the experiments on a 100-core microprocessor than the existing distributed thermal management methods.**

***Index Terms*— Distributed control, dynamic thermal management (DTM), many-core, multicore, task migration.**

I. Introduction

Multicore and upcoming many-core architectures are the trends for current and future microprocessor designs. With the scaling of CMOS devices, the chip multiprocessor is progressing from the multicore era to many-core era [3]–[6] where thousands of cores are predicted to be integrated on a single chip connected by a network-on-chip in the near future. However, the increasing chip complexity and power envelope elevate the peak temperature of chips and increase the on-chip thermal gradients. Thermal constraints are the major driving force for wide adoption of multi/many core architectures. As the power density continues to increase, the excessively high temperature spots would adversely lead to performance degradation, increased cooling costs, reduced reliability, and aging issues for these high-performance chip multiprocessors [14], [15]. Thermal-induced long-term reliability issues such as electromigration, thermal stress migration, time-dependent dielectric breakdown and thermal cycling are especially a growing concern for today's microprocessors as the mean time to failure depends exponentially on the chip temperature due to Arrhenius relationship [2]. The semiconductor industry faces the challenges of maintaining reliability due to the continued increase in die size and number of transistors as well as the constant scaling of transistors for performance [1]. For multi/many core reliability systems, the power consumptions and the resulting temperatures of cores highly depend on the tasks or loads. Therefore, effectively regulating the on-chip temperatures and optimizing related reliability issues of these chips becomes very important and imperative.

Dynamic thermal management (DTM) techniques have been proposed in the past to keep the temperature to stay below a limit [7], [13], [15]. Those techniques, which typically consist of dynamic voltage and frequency scaling (DVFS), task throttling, and clock gating, were first developed for single-core microprocessors. Recently, these techniques have been extended for multicore architectures and multiprocessor system-on-a-chips. They include frequency-control method [23], the combined DVFS and task migration methods [10], [19], the predictive control method [21], [22], and task migration-based methods [16], [20], [21]. However, all these methods are centralized control approaches, which require a central controller (in a core) to monitor the temperature and power of each core in the many-core microprocessor, and globally reallocate the resources and schedule the tasks. Centralized methods may suffer the scalability issue for future many-core systems with hundreds or thousands of computing cores [6]. They also suffer a single point of failure, and large monitoring and communication traffics to the central controller.

To mitigate this problem, distributed thermal control and management techniques have been proposed recently. However, these techniques only consider the steady-state thermal responses and ignore the transient thermal behaviors, which can lead to high peak temperatures in a short period and thus have considerable impacts on the chip reliability. It has been shown that such transient effects will become more significant for high performance chip-multiprocessors with short task execution time when it is close to the thermal time constant of the systems [9]. In this brief, a new task-migration based distributed thermal management is proposed to consider these transient thermal effects to make better task migration decisions to reduce the on-chip temperature variances (TempVar) and hot spots.

We note that DTM typically is applied for temperature-constrained optimization to maximize application performance. However, in this brief, we try to perform the task migration assuming that the highest temperature caused by all the tasks will not exceed the threshold and performance degradation will not be expected. The proposed method can easily combine DVFS and other low power techniques to reduce the temperature as shown in [24]. Even thermal emergencies happen, task migration can still reduce the number of thermal emergencies as we show in experiments, which in turn can lead to less aggressive DFVS and this will translates to the less degraded performance. In addition, hot spot and reduced TempVar will lead to better chip reliability as temperature has exponential impacts on reliability effects such as electromigration and stress migration. Thus, reducing the TempVar and hot spots become an important issue.

The rest of the brief is organized as follows. In Section II, we present the proposed method with the algorithm flow. In Section III, the proposed method is evaluated and its significance is discussed. Section IV concludes this brief.

II. New Distributed Thermal Management Method

A. Effective Initial Temperature for Task Scheduling

Traditional task migration methods typically assign heavy tasks to the cores with low steady-state temperature to balance the on-chip temperature profile [11], [20], [21]. Due to the transient temperature effects, such a simple strategy will not work well as the transient

Manuscript received May 30, 2013; revised October 18, 2013 and December 24, 2013; accepted February 23, 2014. Date of publication March 31, 2014; date of current version January 30, 2015. This work was supported in part by the NSF under Grant CCF-1255899, in part by the Semiconductor Research Corporation (SRC) under Grant 2013-TJ-2417, and in part by the Academic Senate COR Fellowship.

Z. Liu, S. X.-D. Tan, X. Huang are with Department of Electrical Engineering, University of California, Riverside, CA 92521 USA (e-mail: zliu008@ucr.edu; stan@ee.ucr.edu).

H. Wang is with the School of Microelectronics and Solid-State Electronics, University of Electronic Science and Technology of China, Chengdu 610054, China.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2014.2309331

1063-8210 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

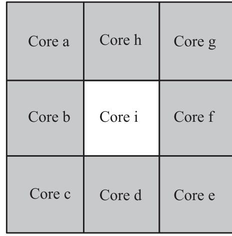

Fig. 1. Local region used to calculate the distributed $T_{\mathrm{eff},N(i)}$ in core i and its adjacent cores.

temperature depends on not only the thermal conductance G, but also thermal capacitance C of the whole processor (including package and heat sinks, etc.). In this brief, to mitigate this problem, we use a new temperature metric, called effective initial temperature, for distributed task migrations considering the influence of transient effects. The effective initial temperature is derived from zeroth frequency domain moment matching as [17], [18]

$$T_{\mathrm{eff}} = G^{-1} C T(t_0) \tag{1}$$

in which $T(t_0)$ is the initial temperature distribution. Thus, T_{eff} represents the thermal response due to the initial temperature $T(t_0)$ under a specific chip structure (including the package) defined by thermal matrices G and C, which incorporates the package related transient thermal effect. In this brief, we demonstrate the effectiveness of T_{eff} in guiding distributed task migration for balancing on-chip temperature profile.

B. Distributed Calculation of Effective Initial Temperature

According to (1), the calculation of each element in T_{eff} requires all elements in $T(t_0)$, which is the global distribution of the temperature values across the whole chip. However, the distributed version of T_{eff} could be obtained in the following way: Instead of using the global temperature distribution and global package model as suggested by (1), we could use a local package model with local temperature values from the current core and its adjacent cores to calculate the distributed effective initial temperature for the local chip region as

$$T_{\mathrm{eff},N(i)} = R_{N(i)} C_{N(i)} T_{N(i)}(t_0) \tag{2}$$

in which $T_{\mathrm{eff},N(i)}$ contains the effective initial temperature values of core i and its adjacent cores in the local region, as shown in Fig. 1. In this figure, $T_{N(i)}$ contains the real temperatures of core i and its adjacent eight cores in the same local region

$$T_{N(i)} = [T_a, T_b, T_c, T_d, T_e, T_f, T_g, T_h, T_i]. \tag{3}$$

$R_{N(i)}$ and $C_{N(i)}$ are the local thermal resistance matrix and thermal capacitance matrix constructed from the elements from row a to i and column a to i from the original G^{-1} and C matrices.

We remark that $T_{\mathrm{eff},N(i)}$ is the localized version of T_{eff} for localized core area, as shown in Fig. 1. In this way, the effective initial temperature can be calculated in a distributed manner in each local region, using the thermal parameters of each local core area and the local temperature distribution of its neighborhood, which systematically considers the impact from its immediate neighbors instead of using ad-hoc-based approaches to consider neighbor effects [16].

C. Proposed New Distributed Task Migration Method

In our proposed approach, we use distributed effective initial temperature as a new thermal metric that takes account of transient thermal effect to guide task migration decision. The new approach

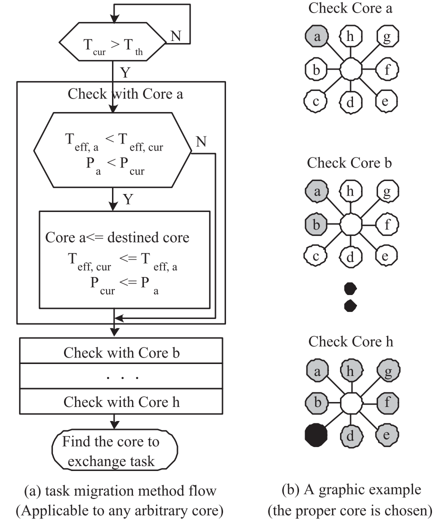

Fig. 2. Proposed distributed task migration method.

is distributed in the sense that task migrations happen only between the two adjacent cores, as shown in Fig. 1, where core *a-h* are the adjacent cores of core i. The proposed distributed task migration method can be outlined as the following: we define the current core as the one that the task migration policy is applied to, the destined core as the adjacent core that could potentially exchange task with the current core if the following two task migration criteria are satisfied.

1) $T_{\mathrm{eff,des}} < T_{\mathrm{eff,cur}}$, where $T_{\mathrm{eff,des}}$ is the computed (in distributed way) effective initial temperature of the destined core using (2) and $T_{\mathrm{eff,cur}}$ is that of the current core.
2) $P_{\mathrm{des}} < P_{\mathrm{cur}}$, where P_{des} is the task load of the destined core in the new execution cycle, and P_{cur} is the counterpart of the current core.

The details of the distributed task migration method flow are shown in Fig. 2(a). The task migration policy will be activated if the temperature of the current core exceeds a certain threshold temperature T_{th}. The current core communicates with its adjacent cores to check if the migration criteria (1) and (2) are satisfied. The flow starts with core a (the upper left adjacent core). If the criteria are satisfied, core a is selected as the destined core for task migration; at the same time, the thermal and load parameters of the current core ($T_{\mathrm{eff,cur}}$, P_{cur}) are updated by the thermal and load parameters of core a ($T_{\mathrm{eff},a}$, P_a,). Then, the flow continues to check with core b, if the migration criteria are satisfied, it indicates that core b is a better migration choice with even lower ($T_{\mathrm{eff},b}$, P_b) than the previous migration choice of the destined core, and thus will be selected to replace the previous choice as the destined core for task migration. The same procedure will be repeated from core c to core h. The flow continues to update the choice and will finally find the best choice for task migration among the eight adjacent cores, which has the lowest value of effective initial temperature (T_{eff}) and load (P). An example flow is shown in Fig. 2(b) in which the circle nodes represent the cores, the solid gray nodes indicate the checked cores, and the solid black node indicates the chosen core for task migration at the end of the task migration method flow. In this example, the new method checks with all of the eight adjacent cores (stamped with gray) and finally selects core c as the destined core (stamped with black) to migrate the task to.

The resulting distributed thermal management method can take account of the task load, the thermal influence from each core and its adjacent region, and the thermal transient effect due to the package properties to more effectively reduce thermal variance across all the cores.

Algorithm 1 Distributed Task Migration Flow

Require: Task loads, many-core processor configuration
Ensure: Optimized temperature distribution
Start simulation at initial package temperature
for each execution cycle **do**
1. Simulate power traces under different task loads.
2. Obtain temperature responses of the cores
if migration criteria is met **then**
Perform migration using the scheme in Fig. 2.
end if
end for

D. Overall Run-Time Thermal Management Scheme

Like [11] and [12], we also assume that each task occupies an equal slice of execution time (one execution cycle), and power traces for each task could be obtained from OS or predicted from the history of the power traces using some estimation methods like time series prediction methods. We assume that the multi/many-core microprocessor is executing different sets of tasks in different execution cycles, and at the beginning of each execution cycle, the proposed distributed task migration policy is activated in each core if the temperature of that core T_{core} exceeds a certain threshold (T_{th}), and task migration between the two adjacent cores is performed if the task migration criteria discussed in Section II-C are satisfied. In this way, the tasks are scheduled to balance the temperature profile of the multi/many-core microprocessors in the new execution cycle. The algorithm flow of the proposed thermal management scheme is summarized in Algorithm 1.

In this brief, we are focusing on how the task configuration could be optimized for more balanced on-chip temperature profile through task migrations with the proposed thermal metric. Also, as in [11], we assume that the migration overhead is small comparing with the task execution cycle, and thus the overheads between the two task execution cycles are negligible.

III. Numerical Results

A. Experiment Setups

The proposed thermal management method is implemented using MATLAB 7.0. Hot spot [15] is used to build the thermal model and simulate the temperature response of a 100-core processor. Each core of the processor has geometry of 4 mm × 4 mm, and the thickness of the processor chip is 0.15 mm. The thickness of heat spreader and heat sink is 1.0 and 6.9 mm, respectively. The thermal conductivity of the chip is 100 W/(mK), and that of the heat spreader and heat sink is 400 W/(mK). The specific heat of the chip is 1.75 J/(m^{3}K), and that of the heat spreader and heat sink is 3.55 J/(m^{3}K). The heat convection capacitance of the heat sink is set to 4600 J/K, and the starting temperature of the chip package is set to be 60 °C. The sampling interval of the thermal data of hot spot is set to be 30 μs to preserve simulation accuracy.

Similar to [11] and [20], we used Wattch as the architecture level power analysis tool [8], and simulate the detailed transient power traces using 16 different dynamic workloads (ammp, apsi, bzip2, equake, galgel, gcc, lucas, mesa, parser, twolf, vpr, applu, art, crafty, fma3d, gap) from SPEC2000 benchmarks. The proposed distributed task migration policy, as discussed in Section II-C will be applied to each core to optimize the distributed task assignment at the beginning of each execution cycle (2048 time steps or 61.44 ms) based on the computed T_{eff} (in distributed way) and task loads, and the simulation consists of 20 task execution cycles in total.

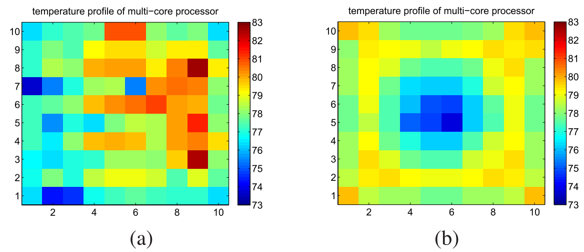

Fig. 3. Temperature distribution across 100 processor cores at the end of 20th task execution cycle. (a) Using real temperature. (b) Using the proposed effective initial temperature.

TABLE I
TempVar and MaxTempDiff

Methods	TempVar	MaxTempDiff
Centralize-Teff	1.242 $^oC^2$	5.961 ^oC
Centralize	1.666 $^oC^2$	8.858 ^oC

B. Task Scheduling Based on the New Effective Initial Temperature

The 16 benchmark tasks with different dynamic workloads are randomly assigned to the cores of the 100-core chip at the beginning of each task execution cycle, and the DTM would be activated once the temperature of the cores exceeds the threshold value of $T_{\text{th}} = 70$ °C.

We first show that the newly proposed temperature metric can lead to more reduction of the TempVar. Fig. 3 shows temperature differences (in °C) across all the 100 cores under these two methods at the end of 20th task execution cycle. When using the real temperature, most of the hot spots appear around the right side of the whole chip and the temperature reach to 83 °C. On the other hand, when using the proposed effective initial temperature, the high temperature spots have been moved to the four corners, with reduced value (about 79 °C) and reduced temperature variations.

The resulting TempVar and maximum temperature differences (MaxTempDiff) at the end of the 20th task execution cycle under different task scheduling scheme are summarized in Table I, in which the notations Centralize-Teff represents the centralized method using the proposed thermal metric T_{eff}; Centralize represents the conventional centralized method using real temperature.

It is interesting to note that the new method automatically tries to find cores with better heat removal capability for heavier tasks. Fig. 4 shows snapshots of the task configurations across the 100-core under different centralized task scheduling scheme at the beginning of the 20th task execution cycle with the thermal management being activated. The tasks with light or heavy workloads are configured across the 100 cores to balance the on-chip temperature profile based on the task migration criteria. Fig. 4(a) is the task profile under the conventional centralized task scheduling method using real temperature while Fig. 4(b) is the task profile under the centralized task scheduling method using the proposed effective initial temperature. The on-chip task configuration differs significantly. Under the conventional method, the task assignment depends only on the temperature of that core, and we can observe that the heavy tasks are getting too concentrated, which will lead to elevated temperature in the new execution cycle. However, when looking at the task configuration under the centralized task scheduling method using the proposed effective initial temperature shown in Fig. 4(b), we could clearly observe that the heavy tasks are separated and moved away from the center to the corner of the core region, where the heat removal is more efficient as those corner regions have shorter thermal paths to

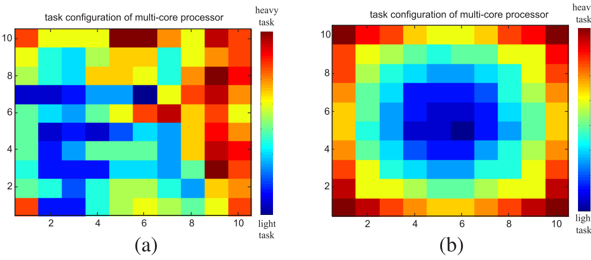

Fig. 4. On-chip task configurations at the beginning of the 20th task execution cycle under centralized task scheduling methods. (a) Using real temperature. (b) Using the proposed effective initial temperature.

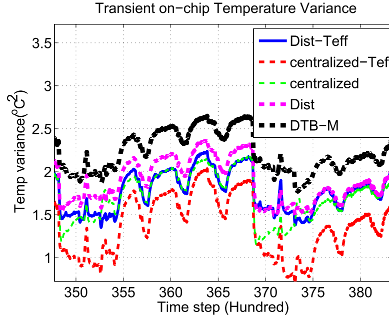

Fig. 5. Comparison of the transient TempVar under different thermal management policy.

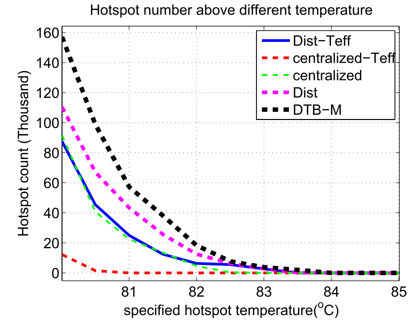

Fig. 6. Occurrence of thermal hot spots above different temperature levels during task execution.

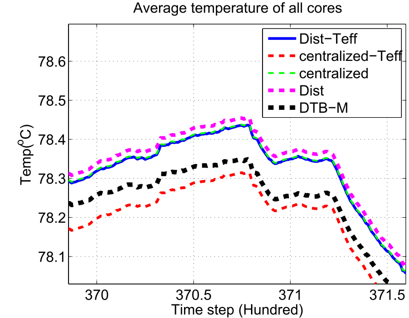

Fig. 7. Comparison of on-chip average temperature during task execution.

heat sink and ambient and less thermal capacity to store the thermal energy for a given initial temperature in our chip configuration.

C. Performance Evaluation of Distributed Thermal Management

We compare the result of the proposed task migration schemes against the conventional ones and the recently reported DTB-M in [12] using the same 16 dynamic workloads of SPEC2000 benchmarks on the 100 core processor platform as in Section III-B. Fig. 5 shows the comparison of transient TempVar, in which the notation Centralize, and Centralize-Teff are the same as in Section III-B, and Dist-Teff represents the distributed method using effective initial temperature, which is also our proposed distributed thermal management method shown by the task scheduling framework in Fig. 2; Dist represents the distributed method using the same framework in Fig. 2, except it uses real temperature instead of T_{eff}; DTB-M is the existing method in [11] and [12].

The centralized method using T_{eff} achieves the best results in term of reducing the on-chip temperature variances among all these methods, while the proposed distributed thermal management method using T_{eff} shows comparable performance with the centralized thermal management method using real temperature. Among the three distributed thermal management method, the proposed one that uses T_{eff} for task migration leads to less TempVar than the counterpart that uses real temperature, and both are more efficient than the existing DTB-M in terms of TempVar reduction.

The reduction of on-chip TempVar could help to reduce the thermal hot spots, the location with temperature over 80 °C. Fig. 6 shows the statistics of thermal hot spot occurrence above different temperature levels during the task execution. The centralized method using T_{eff} achieves the best thermal hot spot reduction, and the proposed distributed method shows comparable reduction of thermal hot spot compared with the centralized method that uses real temperature to guide task migrations. Compared with the alternative distributed counterpart method that uses real temperature, the proposed method leads to 21% reduction of thermal hot spot occurrence during task executions. Compared with the existing DTB-M, the number of thermal hot spots encountered during task executions is reduced by 44% under the proposed distributed task migration method. Thus, as a distributed task migration method, our proposed method can remove on-chip thermal hot spots more effectively compared with other distributed methods.

Fig. 7 shows the comparison of average temperature of the chip employing different thermal management methods. The centralized method with T_{eff} gives the best results. Among the three distributed methods, the DTB-M method does reach the lowest average temperature among the distributed thermal management method, the improvement on the average temperature over other task migration methods, however, is marginal. This figure shows that improving the average temperature may not always lead to better reduction of temperature variations and related hot spots, as shown in Figs. 5 and 6. Hence, these results clearly demonstrate that the effective initial temperature T_{eff} is a better thermal metric compared with real temperature, which lead to more balanced on-chip temperature distribution across all the cores compared with other distributed methods.

IV. CONCLUSION

We have proposed a distributed task migration thermal management technique using a new temperature metric to reduce the on-chip TempVar and thermal hot spots for many-core microprocessors. Experimental results on a 100-core microprocessor have shown that the proposed distributed method works more effectively to reduce the number of thermal hot spots (21% more thermal hot spot reduction compared with the alternative approach under the same distributed framework, and 44% more thermal hot spot reduction compared with the existing distributed thermal management methods). The results also suggested that the proposed method leads to smaller temperature variations across the many-core microprocessor, which will have more positive impacts on the chip reliability effects.

REFERENCES

[1] "Critical reliability challenges for the international technology roadmap for semiconductors (ITRS)," *Int. Sematech Technol. Transf.*, Document 03024377A-TR, 2003.

[2] "Failure mechanisms and models for semiconductor devices," *JEDEC* Publication Document JEP122-A, Jedec Solid State Technol. Assoc., 2002.

[3] *Intel's 48-Core Single-Chip Cloud Computer* [Online]. Available: http://www.intel.com/content/www/us/en/research/intel-labs-single-chip-cloud-overview-paper.html?wapkw=48-core+single

[4] *The Intel Xeon Phi Coprocessor (60 cores and 240 threads and 1teraflops)* [Online]. Available: http://www.intel.com/content/www/us/en/high-performance-computing/high-performance-xeon-phi-coprocessor-brief.html

[5] *Tilera's 64-Core Processor: TILEProf64* [Online]. Available: http://www.tilera.com/products/processors/TILE-Gx_Family.

[6] S. Borkar, "Thousand core chips: A technology perspective," in *Proc. 44th Annu. DAC*, 2007, pp. 746–749.

[7] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *Proc. 7th Int. Symp. High-Perform. Comput. Archit.*, 2001, pp. 171–182.

[8] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. 27th ISCA*, Jun. 2000, pp. 83–94.

[9] T. Chantem, X. S. Hu, and R. P. Dick, "Temperature-aware scheduling and assignment for hard real-time applications on MPSoCs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 10, pp. 1884–1897, Oct. 2011.

[10] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," in *Proc. 33rd Annu. Int. Symp. Comput. Archit.*, Washington, DC, USA, 2006, pp. 78–88.

[11] Y. Ge, Q. Qiu, and Q. Wu, "A multi-agent framework for thermal aware task migration in many-core systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 10, pp. 1758–1771, Oct. 2012.

[12] Y. Ge, P. Malani, and Q. Qiu, "Distributed task migration for thermal management in many-core systems," in *Proc. 47th ACM/IEEE DAC*, Jun. 2010, pp. 579–584.

[13] S. Gunther, F. Binns, D. Carmean, and J. Hall, "Managing the impact of increasing microprocessor power consumption," *Intel Technol. J.*, vol. 5, no. 1, pp. 1–9, Mar. 2001.

[14] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 5, pp. 501–513, May 2006.

[15] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," in *Proc. Int. Symp. Comput. Archit.*, 2003, pp. 2–13.

[16] G. Liu, M. Fan, and G. Quan, "Neigbor-aware dynamic thermal management for multi-core platform," in *Proc. Des., Autom. Test Conf. Eur.*, 2012, pp. 187–192.

[17] P. Liu, H. Li, L. Jin, W. Wu, S. X.-D. Tan, and J. Yang, "Fast thermal simulation for runtime temperature tracking and management," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 12, pp. 2882–2893, Dec. 2006.

[18] Z. Liu, T. Xu, S. X.-D. Tan, and H. Wang, "Dynamic thermal management for multi-core microprocessors considering transient thermal effect," in *Proc. 18th ASPDAC*, Jan. 2013, pp. 473–478.

[19] F. Mulas, D. Atienza, A. Acquaviva, and S. Carta, "Thermal balancing policy for multiprocessor stream computing platforms," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 28, no. 12, pp. 1870–1882, Dec. 2009.

[20] M. Powell, M. Gomaa, and T. N. Vijaykumar, "Heat-and-run: Leveraging SMT and CMP to manage power density through the operating systems," *ACM SIGPLAN Notices*, vol. 39, no. 11, pp. 260–270, 2004.

[21] I. Yeo, C. C. Liu, and E. J. Kim, "Predictive dynamic thermal management for multicore systems," in *Proc. 45th ACM/IEEE DAC*, New York, NY, USA, Jun. 2008, pp. 734–739.

[22] F. Zanini, D. Atienza, L. Benini, and G. De Micheli, "Multicore thermal management with model predictive control," in *Proc. 19th Eur. Conf. Cicuit Theory Des.*, Piscataway, NJ, USA, Aug. 2009, pp. 90–95.

[23] F. Zanini, D. Atienza, and G. De Micheli, "A control theory approach for thermal balancing of MPSoC," in *Proc. ASPDAC*, Jan. 2009, pp. 37–42.

[24] X. Zhou, J. Yang, Y. Xu, Y. Zhang, and J. Zhao, "Thermal-aware task scheduling for 3D multicore processors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 1, pp. 60–71, Jan. 2010.